Anime

April 2, 2022

1 MyAnimeList Data

1.1 Task Complete:

Below are the list that as a group have completed so far since the proposal was due.

- Data pre-processing: Completed
- Linear Regression: Completed
- K-nearest neighbors: Completed
- Squares and Cosine Similarity w/ KNN: Almost complete
- K-Mean Cluster w/ PCs: Almost Complete

1.2 To-Do List: From now until the 18th:

- Apriori
- Analyze Results
- Work on PowerPoint
- Prep for Video presentation
- Project report writeup

1.3 Below are the works that we have done so far

1.4 Data Pre-Processing

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import ast
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
from fuzzywuzzy import process
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import *
from statistics import mean
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-
packages/fuzzywuzzy/fuzz.py:11: UserWarning: Using slow pure-python
```

```
SequenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-
Levenshtein to remove this warning')
```

```
[2]: data = './datasets/anime_data.csv'
```

```
[3]: df = pd.read_csv(data)
```

```
[4]: df.head(5)
```

```
[4]:
   mal_id                 aired_from                   aired_to  \
0       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
1     100  2001-04-04T00:00:00+00:00  2001-06-27T00:00:00+00:00
2    1000  1978-03-14T00:00:00+00:00  1979-02-13T00:00:00+00:00
3   10003  2008-01-01T00:00:00+00:00                        NaN
4   10005  2007-03-31T00:00:00+00:00                        NaN

        duration  episodes                                             genres  \
0  24 min per ep        26  ['Action', 'Adventure', 'Comedy', 'Drama', 'Sc…
1  23 min per ep        13  ['Comedy', 'Drama', 'Fantasy', 'Magic', 'Roman…
2  25 min per ep        42  ['Action', 'Sci-Fi', 'Adventure', 'Space', 'Dr…
3   2 min per ep        15         ['Comedy', 'Dementia', 'Horror', 'Seinen']
4    1 hr 35 min         1         ['Action', 'Adventure', 'Mecha', 'Sci-Fi']

   popularity    premiered     rank                          rating  score  \
0          38  Spring 1998     27.0  R - 17+ (violence & profanity)   8.79
1        2075  Spring 2001   2703.0       PG-13 - Teens 13 or older   7.21
2        2980  Spring 1978   1008.0       PG-13 - Teens 13 or older   7.71
3        6848          NaN  10146.0                R+ - Mild Nudity   5.05
4       10765          NaN   6121.0                    G - All Ages   6.43

   scored_by    source           status  \
0     544987  Original  Finished Airing
1      23787     Manga  Finished Airing
2       7059     Manga  Finished Airing
3       1181  Original  Finished Airing
4        228   Unknown  Finished Airing

                                      studios  \
0         [{'mal_id': 14, 'name': 'Sunrise'}]
1  [{'mal_id': 34, 'name': 'Hal Film Maker'}]
2  [{'mal_id': 18, 'name': 'Toei Animation'}]
3                                          []
4    [{'mal_id': 455, 'name': 'Palm Studio'}]

                                            synopsis  \
0  In the year 2071, humanity has colonized sever…
1  Due to her father's remarriage, robust 16-year…
```

```
2  It is 2977 AD and mankind has become stagnant…
3  In these jokey short films, many of them crude…
4  This theatrical version based on the manga by …

                                   title  \
0                           Cowboy Bebop
1   Shin Shirayuki-hime Densetsu Prétear
2          Uchuu Kaizoku Captain Herlock
3      Kago Shintarou Anime Sakuhin Shuu
4  Tetsujin 28-gou: Hakuchuu no Zangetsu

                           title_english   type
0                           Cowboy Bebop     TV
1  Prétear: The New Legend of Snow White     TV
2           Space Pirate Captain Harlock     TV
3                                    NaN    OVA
4                                    NaN  Movie
```

1.4.1 Extracting studio sequences into a new columns

Source: https://stackoverflow.com/questions/71432733/pandas-extracting-a-phrase-in-a-dict-column?noredirect=1#comment126259925_71432733

In case of the items in the column is just string, convert the column into actual object

[5]:

```
df['studios'] = df['studios'].apply(ast.literal_eval)
```

Implementing .str to access indexes/keys from the lists/dicts of items in a column, and use a combination of pipe and where to fallback to the original values where the result from .str to returns NaN

[6]:

```
df['studios'] = df['studios'].str[0].str['name'].pipe(lambda x: x.where(x.
 ↪notna(), df['studios']))
df.head(5)
```

[6]:

```
   mal_id                 aired_from                   aired_to  \
0       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
1     100  2001-04-04T00:00:00+00:00  2001-06-27T00:00:00+00:00
2    1000  1978-03-14T00:00:00+00:00  1979-02-13T00:00:00+00:00
3   10003  2008-01-01T00:00:00+00:00                        NaN
4   10005  2007-03-31T00:00:00+00:00                        NaN

        duration  episodes                                             genres  \
0  24 min per ep        26  ['Action', 'Adventure', 'Comedy', 'Drama', 'Sc…
1  23 min per ep        13  ['Comedy', 'Drama', 'Fantasy', 'Magic', 'Roman…
2  25 min per ep        42  ['Action', 'Sci-Fi', 'Adventure', 'Space', 'Dr…
3   2 min per ep        15         ['Comedy', 'Dementia', 'Horror', 'Seinen']
4    1 hr 35 min         1         ['Action', 'Adventure', 'Mecha', 'Sci-Fi']
```

```
   popularity    premiered     rank                          rating  score  \
0          38  Spring 1998     27.0  R - 17+ (violence & profanity)   8.79
1        2075  Spring 2001   2703.0       PG-13 - Teens 13 or older   7.21
2        2980  Spring 1978   1008.0       PG-13 - Teens 13 or older   7.71
3        6848          NaN  10146.0                R+ - Mild Nudity   5.05
4       10765          NaN   6121.0                    G - All Ages   6.43

   scored_by    source           status          studios  \
0     544987  Original  Finished Airing          Sunrise
1      23787     Manga  Finished Airing  Hal Film Maker
2       7059     Manga  Finished Airing  Toei Animation
3       1181  Original  Finished Airing               []
4        228   Unknown  Finished Airing     Palm Studio

                                            synopsis  \
0  In the year 2071, humanity has colonized sever…
1  Due to her father's remarriage, robust 16-year…
2  It is 2977 AD and mankind has become stagnant…
3  In these jokey short films, many of them crude…
4  This theatrical version based on the manga by …

                                   title  \
0                           Cowboy Bebop
1   Shin Shirayuki-hime Densetsu Prétear
2          Uchuu Kaizoku Captain Herlock
3      Kago Shintarou Anime Sakuhin Shuu
4  Tetsujin 28-gou: Hakuchuu no Zangetsu

                           title_english   type
0                           Cowboy Bebop     TV
1  Prétear: The New Legend of Snow White     TV
2           Space Pirate Captain Harlock     TV
3                                    NaN    OVA
4                                    NaN  Movie
```

1.4.2 Extract genre list into an individual row

[7]:

```
df['genres'].head(5)
```

[7]:

```
0    ['Action', 'Adventure', 'Comedy', 'Drama', 'Sc…
1    ['Comedy', 'Drama', 'Fantasy', 'Magic', 'Roman…
2    ['Action', 'Sci-Fi', 'Adventure', 'Space', 'Dr…
3           ['Comedy', 'Dementia', 'Horror', 'Seinen']
4           ['Action', 'Adventure', 'Mecha', 'Sci-Fi']
Name: genres, dtype: object
```

Convert the values in the genres column to actual list, because it might just look like

a list but actually be a string.

[8]:
```
df['genres'] = df['genres'].apply(ast.literal_eval)
```

Implementing .explode() for genres column

[9]:
```
data = df.explode('genres').reset_index(drop = True)
```

[10]:
```
data.head(5)
```

[10]:
```
   mal_id                 aired_from                   aired_to  \
0       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
1       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
2       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
3       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
4       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00

        duration  episodes     genres  popularity    premiered  rank  \
0  24 min per ep        26     Action          38  Spring 1998  27.0
1  24 min per ep        26  Adventure          38  Spring 1998  27.0
2  24 min per ep        26     Comedy          38  Spring 1998  27.0
3  24 min per ep        26      Drama          38  Spring 1998  27.0
4  24 min per ep        26     Sci-Fi          38  Spring 1998  27.0

                           rating  score  scored_by    source  \
0  R - 17+ (violence & profanity)   8.79     544987  Original
1  R - 17+ (violence & profanity)   8.79     544987  Original
2  R - 17+ (violence & profanity)   8.79     544987  Original
3  R - 17+ (violence & profanity)   8.79     544987  Original
4  R - 17+ (violence & profanity)   8.79     544987  Original

           status  studios  \
0  Finished Airing  Sunrise
1  Finished Airing  Sunrise
2  Finished Airing  Sunrise
3  Finished Airing  Sunrise
4  Finished Airing  Sunrise

                                            synopsis         title  \
0  In the year 2071, humanity has colonized sever…  Cowboy Bebop
1  In the year 2071, humanity has colonized sever…  Cowboy Bebop
2  In the year 2071, humanity has colonized sever…  Cowboy Bebop
3  In the year 2071, humanity has colonized sever…  Cowboy Bebop
4  In the year 2071, humanity has colonized sever…  Cowboy Bebop

  title_english type
0  Cowboy Bebop   TV
1  Cowboy Bebop   TV
```

```
2  Cowboy Bebop   TV
3  Cowboy Bebop   TV
4  Cowboy Bebop   TV
```

1.4.3 Data Information + Rows and Columns

[11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35984 entries, 0 to 35983
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   mal_id          35984 non-null  int64
 1   aired_from      35977 non-null  object
 2   aired_to        20657 non-null  object
 3   duration        35984 non-null  object
 4   episodes        35984 non-null  int64
 5   genres          35969 non-null  object
 6   popularity      35984 non-null  int64
 7   premiered       13621 non-null  object
 8   rank            33954 non-null  float64
 9   rating          35984 non-null  object
 10  score           35984 non-null  float64
 11  scored_by       35984 non-null  int64
 12  source          35984 non-null  object
 13  status          35984 non-null  object
 14  studios         35984 non-null  object
 15  synopsis        35465 non-null  object
 16  title           35984 non-null  object
 17  title_english   19120 non-null  object
 18  type            35984 non-null  object
dtypes: float64(2), int64(4), object(13)
memory usage: 5.2+ MB
```

[12]:
```
data.shape
```

[12]:
```
(35984, 19)
```

1.4.4 Looking for missing value within the dataset

[13]:
```
data.isnull().sum()
```

[13]:
```
mal_id               0
aired_from           7
aired_to         15327
duration             0
episodes             0
```

```
genres               15
popularity            0
premiered         22363
rank               2030
rating                0
score                 0
scored_by             0
source                0
status                0
studios               0
synopsis            519
title                 0
title_english     16864
type                  0
dtype: int64
```

1.4.5 Extracting Season and Year from primier column to create two new columns

[14]:

```
data[['premiered_season', 'premiered_year']] = data['premiered'].str.
 ↪split(expand = True)
```

[15]:

```
data.head(5)
```

[15]:

```
   mal_id                 aired_from                    aired_to  \
0       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
1       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
2       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
3       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00
4       1  1998-04-03T00:00:00+00:00  1999-04-24T00:00:00+00:00

        duration  episodes     genres  popularity    premiered  rank  \
0  24 min per ep        26     Action          38  Spring 1998  27.0
1  24 min per ep        26  Adventure          38  Spring 1998  27.0
2  24 min per ep        26     Comedy          38  Spring 1998  27.0
3  24 min per ep        26      Drama          38  Spring 1998  27.0
4  24 min per ep        26     Sci-Fi          38  Spring 1998  27.0

                          rating  …  scored_by    source           status  \
0  R - 17+ (violence & profanity)  …     544987  Original  Finished Airing
1  R - 17+ (violence & profanity)  …     544987  Original  Finished Airing
2  R - 17+ (violence & profanity)  …     544987  Original  Finished Airing
3  R - 17+ (violence & profanity)  …     544987  Original  Finished Airing
4  R - 17+ (violence & profanity)  …     544987  Original  Finished Airing

   studios                                          synopsis         title  \
0  Sunrise  In the year 2071, humanity has colonized sever…  Cowboy Bebop
1  Sunrise  In the year 2071, humanity has colonized sever…  Cowboy Bebop
```

```
2  Sunrise  In the year 2071, humanity has colonized sever…  Cowboy Bebop
3  Sunrise  In the year 2071, humanity has colonized sever…  Cowboy Bebop
4  Sunrise  In the year 2071, humanity has colonized sever…  Cowboy Bebop

  title_english type premiered_season premiered_year
0  Cowboy Bebop   TV           Spring           1998
1  Cowboy Bebop   TV           Spring           1998
2  Cowboy Bebop   TV           Spring           1998
3  Cowboy Bebop   TV           Spring           1998
4  Cowboy Bebop   TV           Spring           1998

[5 rows x 21 columns]
```

[16]:

```
data.columns
```

[16]:

```
Index(['mal_id', 'aired_from', 'aired_to', 'duration', 'episodes', 'genres',
       'popularity', 'premiered', 'rank', 'rating', 'score', 'scored_by',
       'source', 'status', 'studios', 'synopsis', 'title', 'title_english',
       'type', 'premiered_season', 'premiered_year'],
      dtype='object')
```

1.4.6 Dropping Columns

[17]:

```
data.drop(['mal_id', 'aired_from', 'aired_to', 'synopsis', 'status'], axis = 1, inplace = True)
```

[18]:

```
data.head(5)
```

[18]:

```
        duration  episodes     genres  popularity    premiered  rank  \
0  24 min per ep        26     Action          38  Spring 1998  27.0
1  24 min per ep        26  Adventure          38  Spring 1998  27.0
2  24 min per ep        26     Comedy          38  Spring 1998  27.0
3  24 min per ep        26      Drama          38  Spring 1998  27.0
4  24 min per ep        26     Sci-Fi          38  Spring 1998  27.0

                           rating  score  scored_by    source  studios  \
0  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
1  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
2  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
3  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
4  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise

          title title_english type premiered_season premiered_year
0  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
1  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
2  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
3  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
```

```
4  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
```

Drop primiered column

[19]:
```
data.drop(['premiered'], axis = 1, inplace = True)
```

[20]:
```
data.head(5)
```

[20]:
```
        duration  episodes     genres  popularity  rank  \
0  24 min per ep        26     Action          38  27.0
1  24 min per ep        26  Adventure          38  27.0
2  24 min per ep        26     Comedy          38  27.0
3  24 min per ep        26      Drama          38  27.0
4  24 min per ep        26     Sci-Fi          38  27.0

                           rating  score  scored_by    source  studios  \
0  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
1  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
2  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
3  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
4  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise

          title title_english type premiered_season premiered_year
0  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
1  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
2  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
3  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
4  Cowboy Bebop  Cowboy Bebop   TV           Spring           1998
```

Drop English title column

[21]:
```
data.drop(['title_english'], axis = 1, inplace = True)
```

[22]:
```
data.head(5)
```

[22]:
```
        duration  episodes     genres  popularity  rank  \
0  24 min per ep        26     Action          38  27.0
1  24 min per ep        26  Adventure          38  27.0
2  24 min per ep        26     Comedy          38  27.0
3  24 min per ep        26      Drama          38  27.0
4  24 min per ep        26     Sci-Fi          38  27.0

                           rating  score  scored_by    source  studios  \
0  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
1  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
2  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
3  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
4  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
```

```
          title type premiered_season premiered_year
0  Cowboy Bebop   TV           Spring           1998
1  Cowboy Bebop   TV           Spring           1998
2  Cowboy Bebop   TV           Spring           1998
3  Cowboy Bebop   TV           Spring           1998
4  Cowboy Bebop   TV           Spring           1998
```

1.4.7 Fill NaN with 0 or make the empty column as string

[23]:

```
data['rank'] = data['rank'].fillna(data['rank'].dropna().mode().values[0])
data['premiered_year'] = data['premiered_year'].fillna(data['premiered_year'].
 ↪dropna().mode().values[0])
data['genres'].fillna('', inplace = True)
data['premiered_season'].fillna('', inplace = True)
data.isnull().sum()
```

[23]:

```
duration            0
episodes            0
genres              0
popularity          0
rank                0
rating              0
score               0
scored_by           0
source              0
studios             0
title               0
type                0
premiered_season    0
premiered_year      0
dtype: int64
```

[24]:

```
data.head(3)
```

[24]:

```
        duration  episodes     genres  popularity  rank  \
0  24 min per ep        26     Action          38  27.0
1  24 min per ep        26  Adventure          38  27.0
2  24 min per ep        26     Comedy          38  27.0

                           rating  score  scored_by    source  studios  \
0  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
1  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
2  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise

          title type premiered_season premiered_year
0  Cowboy Bebop   TV           Spring           1998
1  Cowboy Bebop   TV           Spring           1998
```

```
2  Cowboy Bebop    TV            Spring          1998
```

1.4.8 Preprocess User Datat

Following are the code used to preprocess the user_score_data.csv which is originally derived from user_data.csv. This section was commented out and data was exported into a csv since it takes a while to execute.

```
[25]: # user_df = pd.read_csv('./datasets/user_data.csv')
      # user_df.insert(0, 'user_id', range(1, 1 + len(user_df)))
      # user_watched = user_df[['user_id', 'watched']]

      # import ast
      # user_data = []

      # for i in range(len(user_df)):
      #     row = user_watched.iloc[i].watched
      #     row = row.strip('][').split('}, ')
      #     for item in row:
      #         row_dict = {}
      #         if (item[-1] != "}"):
      #             item = item + "}"
      #         item_dict = ast.literal_eval(item)
      #         row_dict['user_id'] = user_watched.iloc[i].user_id
      #         row_dict['mal_id'] = item_dict['mal_id']
      #         row_dict['rating'] = item_dict['score']
      #         user_data.append(row_dict)

      # df_user_data = pd.DataFrame(user_data)
      # df_user_data.to_csv('user_score_data')
```

1.4.9 Linear Regression

```
[26]: user_data_df = pd.read_csv('./datasets/user_score_data.csv',␣
       ↪usecols=['user_id', 'mal_id', 'rating'], dtype={'user_id':'int32', 'mal_id':
       ↪'int32', 'rating':'float32'})
      animes_df = pd.read_csv('./datasets/anime_data.csv', usecols=['mal_id',␣
       ↪'title'], dtype={'mal_id':'int32', 'title':'string'})
```

Not all users will rate every anime. Therefore, there are missing data in the ratings of animes. To have a better prediction, linear regression can be used to generate predictions of missing data based on existing values.

```
[27]: def getOverallUserAvgAnimeRating(user_data_df):
          average = user_data_df.groupby('mal_id')['rating'].agg('mean')
          return pd.DataFrame({'mal_id':average.index, 'rating':average.values})
```

[28]:

```
def getTestTrainData(y):
    test_data = y[y['rating_y'].isna()]
    train_data = y.dropna(subset=['rating_y'])

    y_train = train_data['rating_y']
    X_train = train_data.drop('rating_y', axis=1)

    return test_data, train_data, y_train, X_train
```

[29]:

```
def fillMissingRatingDataLinReg(y):
    test_data, train_data, y_train, X_train = getTestTrainData(y)
    lin_model = LinearRegression().fit(X_train, y_train)

    X_test = test_data.drop('rating_y', axis=1)
    y_pred = lin_model.predict(X_test)

    test_data.loc[test_data.rating_y.isna(), 'rating_y'] = y_pred

    new = pd.concat([test_data, train_data], axis=0).sort_values(by=['mal_id'],␣
 ↪ascending=True)
    new.rename(columns={'rating_y':'rating'}, inplace=True)

    return new
```

[30]:

```
def getComprehensiveUserRating(user_data_df, user_id):
    '''
        Takes user data and fills missing data based on linear regression
        using collaborative average anime rating. Predicts what user of␣
 ↪specified
        id will rate each anime.
    '''
    # get average anime rating
    avg_df = getOverallUserAvgAnimeRating(user_data_df)

    # get all user rating
    y = (user_data_df[user_data_df['user_id'] == user_id])
    y = y.drop(columns=['user_id'])

    merged_y = pd.merge(avg_df, y, on='mal_id',how='left').
 ↪drop(columns=['rating_x'])

    comprehensive_df = fillMissingRatingDataLinReg(merged_y)

    return comprehensive_df
```

[31]:

```
# new = getComprehensiveUserRating(user_data_df, 1)
```

1.4.10 K-Nearest Neighbors

K-nearest neighbors can be used to generate recommendation based on specified anime. Using collaborative filtering, k-nearest neighbors will search for what other animes were enjoyed by other users who also enjoyed watching the specified anime.

[32]:
```
pip install fuzzywuzzy
```

```
Requirement already satisfied: fuzzywuzzy in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(0.18.0)
WARNING: You are using pip version 22.0.3; however, version 22.0.4 is
available.
You should consider upgrading via the
'/Library/Frameworks/Python.framework/Versions/3.8/bin/python3 -m pip install
--upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[33]:
```
animes_users = user_data_df.pivot(index='mal_id', columns='user_id',␣
 ↪values='rating').fillna(0)
animes_users_mat = csr_matrix(animes_users.values)
```

[34]:
```
model_knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20)
model_knn.fit(animes_users_mat)
```

[34]:
```
NearestNeighbors(algorithm='brute', metric='cosine', n_neighbors=20)
```

[35]:
```
def getRecommendations(movie_title, data_matrix, animes_df, model_knn,␣
 ↪n_recommendations):
    model_knn.fit(data_matrix)
    anime_index = process.extractOne(movie_title, animes_df['title'])[2]
    distances, indices = model_knn.kneighbors(data_matrix[anime_index],␣
 ↪n_neighbors=n_recommendations)
    for i in indices:
        print(animes_df['title'][i].where(i != anime_index))

```

[36]:
```
getRecommendations('Bleach', animes_users_mat, animes_df, model_knn, 5)
```

```
3990                                                 <NA>
6198    Iizuka-senpai x Blazer: Ane Kyun! yori The Ani…
5435                              Kanashimi no Belladonna
3093    New Mobile Report Gundam Wing: Frozen Teardrop…
3295                                       Plastic Little
Name: title, dtype: string
```

1.4.11 PCA

```
[37]: data.head(2)
```

```
[37]:         duration  episodes     genres  popularity  rank  \
      0  24 min per ep        26     Action          38  27.0
      1  24 min per ep        26  Adventure          38  27.0

                                 rating  score  scored_by    source  studios  \
      0  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise
      1  R - 17+ (violence & profanity)   8.79     544987  Original  Sunrise

                title type premiered_season premiered_year
      0  Cowboy Bebop   TV           Spring           1998
      1  Cowboy Bebop   TV           Spring           1998
```

```
[38]: features = ['episodes', 'popularity', 'rank', 'score', 'premiered_year']
```

```
[39]: X = data.loc[:, features].values
      y = data.loc[:, ['title']].values
```

```
[40]: X = StandardScaler().fit_transform(X)
```

```
[41]: pca_df = pd.DataFrame(data = X, columns = features).head()
      pca_df.head(3)
```

```
[41]:    episodes  popularity      rank     score  premiered_year
      0  0.335596   -1.449592 -1.527175  2.375764       -1.764233
      1  0.335596   -1.449592 -1.527175  2.375764       -1.764233
      2  0.335596   -1.449592 -1.527175  2.375764       -1.764233
```

```
[42]: projection_pca = PCA(n_components = 5)
```

```
[43]: components = projection_pca.fit_transform(X)
```

```
[44]: two_d = df2 = pd.DataFrame(components, columns = ['Component 1', 'Component 2',␣
       ↪'Component 3', 'Component 4', 'Component 5'])
```

```
[45]: final_df = pd.concat([two_d, data[['title']]], axis = 1)
      final_df.head()
```

```
[45]:    Component 1  Component 2  Component 3  Component 4  Component 5  \
      0     3.305986     0.967371    -1.057385    -0.067117    -0.561301
      1     3.305986     0.967371    -1.057385    -0.067117    -0.561301
      2     3.305986     0.967371    -1.057385    -0.067117    -0.561301
      3     3.305986     0.967371    -1.057385    -0.067117    -0.561301
      4     3.305986     0.967371    -1.057385    -0.067117    -0.561301
```

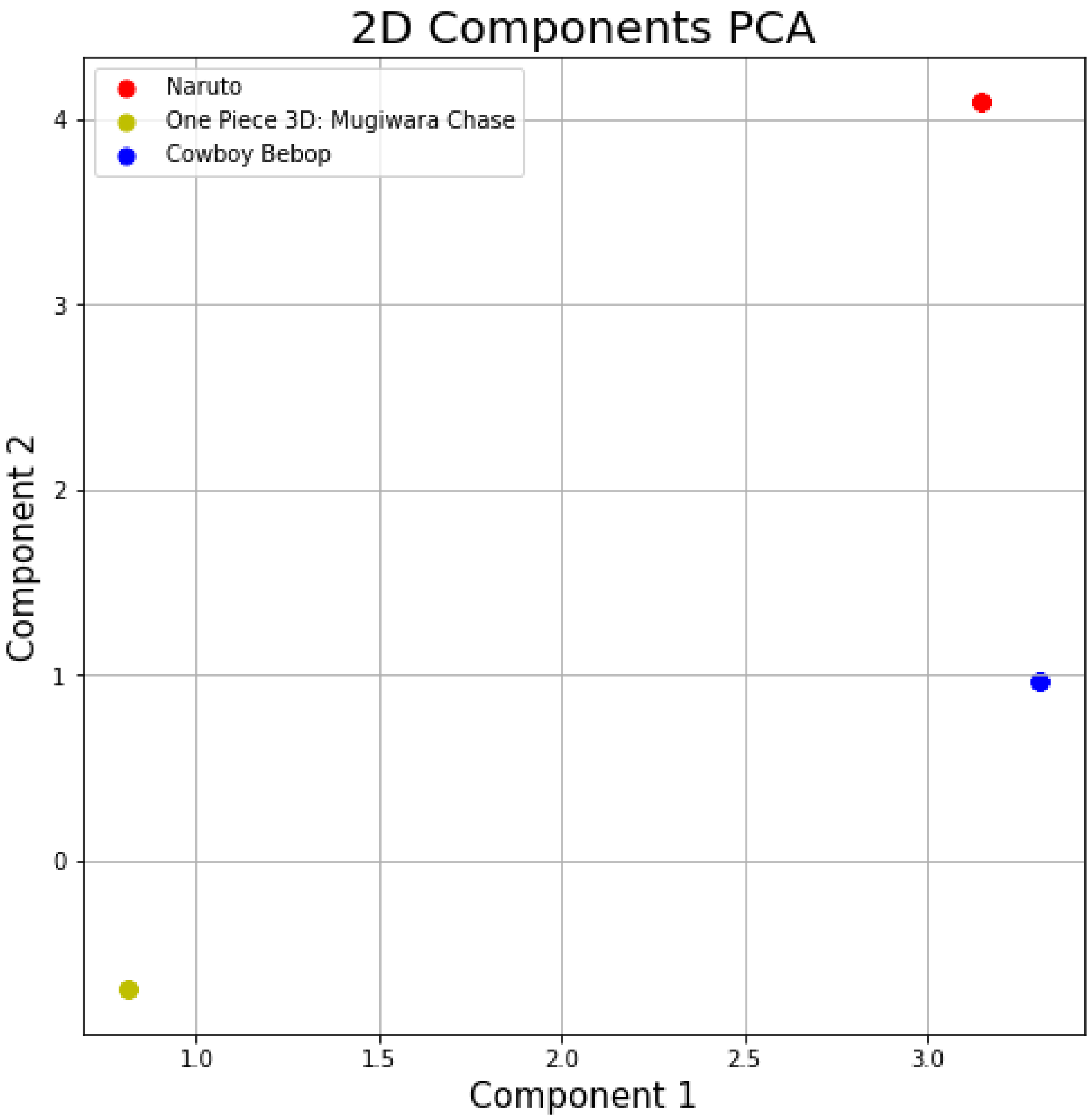
```
          title
0  Cowboy Bebop
1  Cowboy Bebop
2  Cowboy Bebop
3  Cowboy Bebop
4  Cowboy Bebop
```

```
[46]: data['title']
```

```
[46]: 0                        Cowboy Bebop
      1                        Cowboy Bebop
      2                        Cowboy Bebop
      3                        Cowboy Bebop
      4                        Cowboy Bebop
                           …
      35979    One Piece 3D: Mugiwara Chase
      35980    One Piece 3D: Mugiwara Chase
      35981    One Piece 3D: Mugiwara Chase
      35982    One Piece 3D: Mugiwara Chase
      35983    One Piece 3D: Mugiwara Chase
      Name: title, Length: 35984, dtype: object
```

```
[47]: fig = plt.figure(figsize = (8,8))
      ax = fig.add_subplot(1,1,1)
      ax.set_xlabel('Component 1', fontsize = 15)
      ax.set_ylabel('Component 2', fontsize = 15)
      ax.set_title('2D Components PCA', fontsize = 20)

      targets = ['Naruto', 'One Piece 3D: Mugiwara Chase', 'Cowboy Bebop']
      colors = ['r', 'y', 'b']
      for target, color in zip(targets, colors):
          indicesToKeep = final_df['title'] == target
          ax.scatter(final_df.loc[indicesToKeep, 'Component 1'], final_df.
       ↪loc[indicesToKeep, 'Component 2'] , c = color, s = 50)
      ax.legend(targets)
      ax.grid()
```



[ ]:

```

```

1.5 Matrix Factorization - Singular Value Decomposition (SVD)

Followed this tutorial https://towardsdatascience.com/how-did-we-build-book-recommender-systems-in-an-hour-part-2-k-nearest-neighbors-and-matrix-c04b3c2ef55c#:~:text=kNN%20is%20a%20machine%20learning,of%20top%2Dk%20nearest%20neighbors

[48]:

```
# Imports and process needed datasets
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
```

```
import sklearn
from sklearn.decomposition import TruncatedSVD

user_rating_data = './datasets/user_score_data.csv'
df = pd.read_csv(user_rating_data)
user_rating_df = df[['user_id', 'mal_id', 'rating']].copy()

anime_info_data = './datasets/anime_data.csv'
anime_df = pd.read_csv(anime_info_data)
columns = ['aired_from', 'aired_to', 'duration', 'episodes', 'genres',␣
 ↪'popularity', 'premiered', 'rank', 'rating', 'score', 'scored_by', 'source',␣
 ↪'status', 'studios', 'synopsis', 'title', 'type']
anime_df = anime_df.drop(columns, axis=1)
anime_df = anime_df.dropna()
```

1.5.1 Combine datasets and group by title to get total rating count for each show

[49]:
```
combine_user_anime = pd.merge(user_rating_df, anime_df, on='mal_id')
total_ratings = (combine_user_anime.
                     groupby(by = ['title_english'])['rating'].
                 count().
                 reset_index().
                 rename(columns = {'rating' : 'totalRatingCount'})
                 [['title_english', 'totalRatingCount']]
                 )
total_ratings.head()
```

[49]:
```
         title_english  totalRatingCount
0    "Parade" de Satie                14
1              "Star"t                15
2   -OutsideR:RequieM-                17
3           .Koni-chan                 9
4  .hack//G.U. Trilogy                49
```

1.5.2 Narrow the dataset down to anime that have been rated a certain number of times (based on the rating stats)

[50]:
```
userRatings_with_totalRatings = combine_user_anime.merge(total_ratings,␣
 ↪left_on='title_english', right_on='title_english')
userRatings_with_totalRatings.head(40)

popularity_threshold = 100 # this can be changed to narrow the scope of our data
ratings_top_anime = userRatings_with_totalRatings.query('totalRatingCount >=␣
 ↪@popularity_threshold')
n = len(pd.unique(ratings_top_anime['title_english']))
print("Number of unique anime to be used: ", n)
```

```
Number of unique anime to be used:  1710
```

1.5.3 Convert to 2D Matrix and transpose

[51]:
```
ratings_top_anime_pivot = ratings_top_anime.pivot_table(index = 'user_id', columns='title_english', values='rating', aggfunc=np.sum).fillna(0)
transposed_ratings = ratings_top_anime_pivot.values.T
ratings_top_anime_pivot.head()
```

[51]:
```
title_english  .hack//Sign  07-Ghost  11eyes  5 Centimeters Per Second  \
user_id
1                      0.0       0.0     0.0                      10.0
2                      0.0       0.0     9.0                       8.0
3                      0.0       0.0     0.0                       7.0
4                      0.0       6.0     0.0                       0.0
5                      0.0       0.0     0.0                       0.0

title_english  7 Seeds  91 Days  91 Days: Brief Candle  \
user_id
1                  0.0      0.0                    0.0
2                  0.0      9.0                    0.0
3                  0.0      8.0                    0.0
4                  0.0      0.0                    0.0
5                  0.0      8.0                    0.0

title_english  91 Days: Shoal of Time/All Our Yesterdays/Tomorrow and Tomorrow
\
user_id
1                                                            0.0
2                                                            6.0
3                                                            0.0
4                                                            0.0
5                                                            0.0

title_english  A Bridge to the Starry Skies  A Centaur's Life  ...  \
user_id                                                        ...
1                                       0.0               0.0  ...
2                                       0.0               0.0  ...
3                                       0.0               0.0  ...
4                                       0.0               0.0  ...
5                                       0.0               0.0  ...

title_english  the Garden of sinners Chapter 2: Murder Speculation Part A  \
user_id
1                                                            0.0
2                                                            0.0
3                                                            0.0
```

```
4                                                             0.0
5                                                             0.0

title_english  the Garden of sinners Chapter 3: Remaining Sense of Pain  \
user_id
1                                                             0.0
2                                                             0.0
3                                                             0.0
4                                                             0.0
5                                                             0.0

title_english  the Garden of sinners Chapter 4: The Hollow Shrine  \
user_id
1                                                             0.0
2                                                             0.0
3                                                             0.0
4                                                             0.0
5                                                             0.0

title_english  the Garden of sinners Chapter 5: Paradox Paradigm  \
user_id
1                                                             0.0
2                                                             0.0
3                                                             0.0
4                                                             0.0
5                                                             0.0

title_english  the Garden of sinners Chapter 6: Oblivion Recording  \
user_id
1                                                             0.0
2                                                             0.0
3                                                             0.0
4                                                             0.0
5                                                             0.0

title_english  the Garden of sinners Chapter 7: Murder Speculation Part B  \
user_id
1                                                             0.0
2                                                             0.0
3                                                             0.0
4                                                             0.0
5                                                             0.0

title_english  the Garden of sinners Chapter 8: The Final Chapter  \
user_id
1                                                             0.0
2                                                             0.0
```

```
3                                                            0.0
4                                                            0.0
5                                                            0.0

title_english  the Garden of sinners Remix -Gate of seventh heaven-  \
user_id
1                                                            0.0
2                                                            0.0
3                                                            0.0
4                                                            0.0
5                                                            0.0

title_english  tsuritama  xxxHOLiC
user_id
1                    0.0       0.0
2                    9.0       9.0
3                    0.0       0.0
4                    8.0       0.0
5                    0.0       0.0

[5 rows x 1710 columns]
```

1.5.4 Run SVD and calculate Pearson R Correlation Coefficient, (need to figure out num of latent variables for later)

[52]:

```
import warnings
warnings.filterwarnings("ignore", category = RuntimeWarning)

# SVD
SVD = TruncatedSVD(n_components=12, random_state=17)
matrix = SVD.fit_transform(transposed_ratings)

# Correlation Coefficient
corr = np.corrcoef(matrix)
corr.shape
```

[52]:

```
(1710, 1710)
```

1.5.5 Recommendations based on SVD - Random Choice

[53]:

```
anime_titles = ratings_top_anime_pivot.columns
anime_titles_list = list(anime_titles)

# Pick random anime
title_chosen = np.random.choice(anime_titles_list)
print('Recommendations for: ', title_chosen)
```

```
# Get its index and correlation coefficient
title_index = anime_titles_list.index(title_chosen)
corr_title = corr[title_index]

# List the correlated titles with the random title chosen
list(anime_titles[(corr_title<1.0) & (corr_title>0.9)])
```

```
Recommendations for:  Skip Beat!
```

```
[53]: ['Big Windup!',
       'Earl and Fairy',
       'Fruits Basket',
       'Ghost Hunt',
       'Gravitation',
       'Hakuoki ~Demon of the Fleeting Blossom~',
       'Hakuoki ~Demon of the Fleeting Blossom~ Record of the Jade Blood',
       'Hal',
       'ItaKiss',
       'Kamisama Kiss',
       'Kimi ni Todoke - From Me To You Season 2 - A Crush',
       'Kimi ni Todoke: From Me To You 2',
       'Kobato.',
       'Loveless',
       'Lovely Complex',
       "Natsume's Book of Friends",
       "Natsume's Book of Friends Season 2",
       "Natsume's Book of Friends Season 3",
       "Natsume's Book of Friends Season 4",
       'Natsuyuki Rendezvous',
       'No. 6',
       'Ouran High School Host Club',
       'PandoraHearts',
       'Paradise Kiss',
       'Princess Jellyfish',
       'Psychic Detective Yakumo',
       'Special A (S.A)',
       'The Seven Metamorphoses of Yamato Nadeshiko',
       'Vampire Knight',
       'Vampire Knight: Guilty',
       'You and Me 2',
       'You and Me.',
       'Zakuro',
       'tsuritama']
```

1.5.6 Recommendations based on SVD - Input Title

[54]:

```python
# Type in title
title_chosen = "Snow White with the Red Hair"
print('Recommendations for: ', title_chosen, '\n')

# Get its index and correlation coefficient
title_index = anime_titles_list.index(title_chosen)
corr_title = corr[title_index]

# List the correlated titles with the random title chosen
list(anime_titles[(corr_title<1.0) & (corr_title>0.9)])
```

```
Recommendations for:  Snow White with the Red Hair
```

[54]:

```
['A Lull in the Sea',
 'Anonymous Noise',
 'Aoharu x Machinegun',
 'Beyond the Boundary',
 'Blue Spring Ride',
 "I've Always Liked You",
 'Kiss Him, Not Me!',
 'Kiznaiver',
 'Maid Sama!',
 "Monthly Girls' Nozaki-kun",
 'My Little Monster',
 'My Love Story!!',
 'Orange',
 'Prince of Stride: Alternative',
 'Rainbow Days',
 'ReLIFE',
 'Say "I Love You".',
 'Snow White with the Red Hair 2',
 'The Anthem of the Heart',
 'The Lost Village',
 'The World is Still Beautiful',
 'Welcome to the Ballroom',
 'Wolf Girl & Black Prince',
 'Yona of the Dawn']
```

FAY'S TO DO: figure out the right latent variable number, see if we can rank the recommendations list and keep it to 10 recs, see if I can check for accuracy and comparisons for analysis