movie_recommendation_system

November 20, 2023

[90]:
```
!pip install surprise
```

```
Requirement already satisfied: surprise in /usr/local/lib/python3.10/dist-
packages (0.1)
Requirement already satisfied: scikit-surprise in
/usr/local/lib/python3.10/dist-packages (from surprise) (1.1.3)
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-surprise->surprise) (1.3.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-
packages (from scikit-surprise->surprise) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-
packages (from scikit-surprise->surprise) (1.11.3)
```

[155]:
```
# All the Required Libraries
# These Libraries deals with Data Preprocessing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import seaborn as sns
# These Libraries deal with model importation training and Prediction
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from surprise import accuracy
from surprise import KNNBasic
from surprise.model_selection import KFold
```

[92]:
```
# Loading all the sub datasets to be used
sub_set1 = pd.read_csv('/content/tmdb_5000_credits.csv')
sub_set2 = pd.read_csv('/content/tmdb_5000_movies.csv')
sub_set3 = pd.read_csv('/content/ratings_small.csv')
```

Viewing The Initial Rows of the Sub Datasets

[93]:
```
# Viewing the Initial 5 rows of the  Movie credits
sub_set1.head()
```

[93]:
```
   movie_id                                     title  \
0     19995                                    Avatar
1       285  Pirates of the Caribbean: At World's End
2    206647                                   Spectre
3     49026                     The Dark Knight Rises
4     49529                               John Carter

                                                cast  \
0  [{"cast_id": 242, "character": "Jake Sully", "…
1  [{"cast_id": 4, "character": "Captain Jack Spa…
2  [{"cast_id": 1, "character": "James Bond", "cr…
3  [{"cast_id": 2, "character": "Bruce Wayne / Ba…
4  [{"cast_id": 5, "character": "John Carter", "c…

                                                crew
0  [{"credit_id": "52fe48009251416c750aca23", "de…
1  [{"credit_id": "52fe4232c3a36847f800b579", "de…
2  [{"credit_id": "54805967c3a36829b5002c41", "de…
3  [{"credit_id": "52fe4781c3a36847f81398c3", "de…
4  [{"credit_id": "52fe479ac3a36847f813eaa3", "de…
```

[94]:
```
# Viewing the Initial 5 rows of the  Movie dataset
sub_set2.head()
```

[94]:
```
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "…
2  245000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…
3  250000000  [{"id": 28, "name": "Action"}, {"id": 80, "nam…
4  260000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…

                                      homepage      id  \
0                  http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/    285
2   http://www.sonypictures.com/movies/spectre/  206647
3            http://www.thedarkknightrises.com/   49026
4          http://movies.disney.com/john-carter   49529

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":…                en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na…                en
2  [{"id": 470, "name": "spy"}, {"id": 818, "name…                en
3  [{"id": 849, "name": "dc comics"}, {"id": 853,…                en
4  [{"id": 818, "name": "based on novel"}, {"id":…                en
```

```
                          original_title  \
0                                   Avatar
1  Pirates of the Caribbean: At World's End
2                                  Spectre
3                    The Dark Knight Rises
4                              John Carter

                                            overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di…  150.437577
1  Captain Barbossa, long believed to be dead, ha…  139.082615
2  A cryptic message from Bond’s past sends him o…  107.376788
3  Following the death of District Attorney Harve…  112.312950
4  John Carter is a war-weary, former military ca…   43.926995

                             production_companies  \
0  [{"name": "Ingenious Film Partners", "id": 289…
1  [{"name": "Walt Disney Pictures", "id": 2}, {"…
2  [{"name": "Columbia Pictures", "id": 5}, {"nam…
3  [{"name": "Legendary Pictures", "id": 923}, {"…
4        [{"name": "Walt Disney Pictures", "id": 2}]

                             production_countries release_date     revenue  \
0  [{"iso_3166_1": "US", "name": "United States o…   2009-12-10  2787965087
1  [{"iso_3166_1": "US", "name": "United States o…   2007-05-19   961000000
2  [{"iso_3166_1": "GB", "name": "United Kingdom"…   2015-10-26   880674609
3  [{"iso_3166_1": "US", "name": "United States o…   2012-07-16  1084939099
4  [{"iso_3166_1": "US", "name": "United States o…   2012-03-07   284139100

   runtime                                   spoken_languages    status  \
0    162.0  [{"iso_639_1": "en", "name": "English"}, {"iso…  Released
1    169.0           [{"iso_639_1": "en", "name": "English"}]  Released
2    148.0  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},…  Released
3    165.0           [{"iso_639_1": "en", "name": "English"}]  Released
4    132.0           [{"iso_639_1": "en", "name": "English"}]  Released

                                          tagline  \
0                        Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                              A Plan No One Escapes
3                                    The Legend Ends
4             Lost in our world, found in another.

                                     title  vote_average  vote_count
0                                   Avatar           7.2       11800
1  Pirates of the Caribbean: At World's End           6.9        4500
2                                  Spectre           6.3        4466
```

```
3                    The Dark Knight Rises          7.6        9106
4                              John Carter          6.1        2124
```

[95]:
```
# Viewing the Initial 5 rows of the  Movie credits using a Reader Object
reader = Reader()
sub_set3.head()
```

[95]:
```
   userId  movieId  rating   timestamp
0       1       31     2.5  1260759144
1       1     1029     3.0  1260759179
2       1     1061     3.0  1260759182
3       1     1129     2.0  1260759185
4       1     1172     4.0  1260759205
```

Exploratory Data Analysis

[96]:
```
# Renaming the Columns of  credits  dataset inorder to merge it with Movie ␣
 ↪using common column id
sub_set1.columns = ['id','tile','cast','crew']
sub_set2= sub_set2.merge(sub_set1,on='id')
```

[97]:
```
# viewing the improved movies dataset
sub_set2.head()
```

[97]:
```
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "…
2  245000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…
3  250000000  [{"id": 28, "name": "Action"}, {"id": 80, "nam…
4  260000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…

                                       homepage      id  \
0                   http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3            http://www.thedarkknightrises.com/   49026
4          http://movies.disney.com/john-carter   49529

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":…                en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na…                en
2  [{"id": 470, "name": "spy"}, {"id": 818, "name…                en
3  [{"id": 849, "name": "dc comics"}, {"id": 853,…                en
4  [{"id": 818, "name": "based on novel"}, {"id":…                en

                             original_title  \
0                                    Avatar
```

```
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                               John Carter

                                            overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di…  150.437577
1  Captain Barbossa, long believed to be dead, ha…  139.082615
2  A cryptic message from Bond’s past sends him o…  107.376788
3  Following the death of District Attorney Harve…  112.312950
4  John Carter is a war-weary, former military ca…   43.926995

                                production_companies  … runtime  \
0  [{"name": "Ingenious Film Partners", "id": 289…  …   162.0
1  [{"name": "Walt Disney Pictures", "id": 2}, {"…  …   169.0
2  [{"name": "Columbia Pictures", "id": 5}, {"nam…  …   148.0
3  [{"name": "Legendary Pictures", "id": 923}, {"…  …   165.0
4        [{"name": "Walt Disney Pictures", "id": 2}]  …   132.0

                                    spoken_languages    status  \
0  [{"iso_639_1": "en", "name": "English"}, {"iso…  Released
1           [{"iso_639_1": "en", "name": "English"}]  Released
2  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},…  Released
3           [{"iso_639_1": "en", "name": "English"}]  Released
4           [{"iso_639_1": "en", "name": "English"}]  Released

                                          tagline  \
0                     Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                            A Plan No One Escapes
3                                  The Legend Ends
4           Lost in our world, found in another.

                                      title vote_average vote_count  \
0                                    Avatar          7.2      11800
1  Pirates of the Caribbean: At World's End          6.9       4500
2                                   Spectre          6.3       4466
3                     The Dark Knight Rises          7.6       9106
4                               John Carter          6.1       2124

                                       tile  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                               John Carter
```

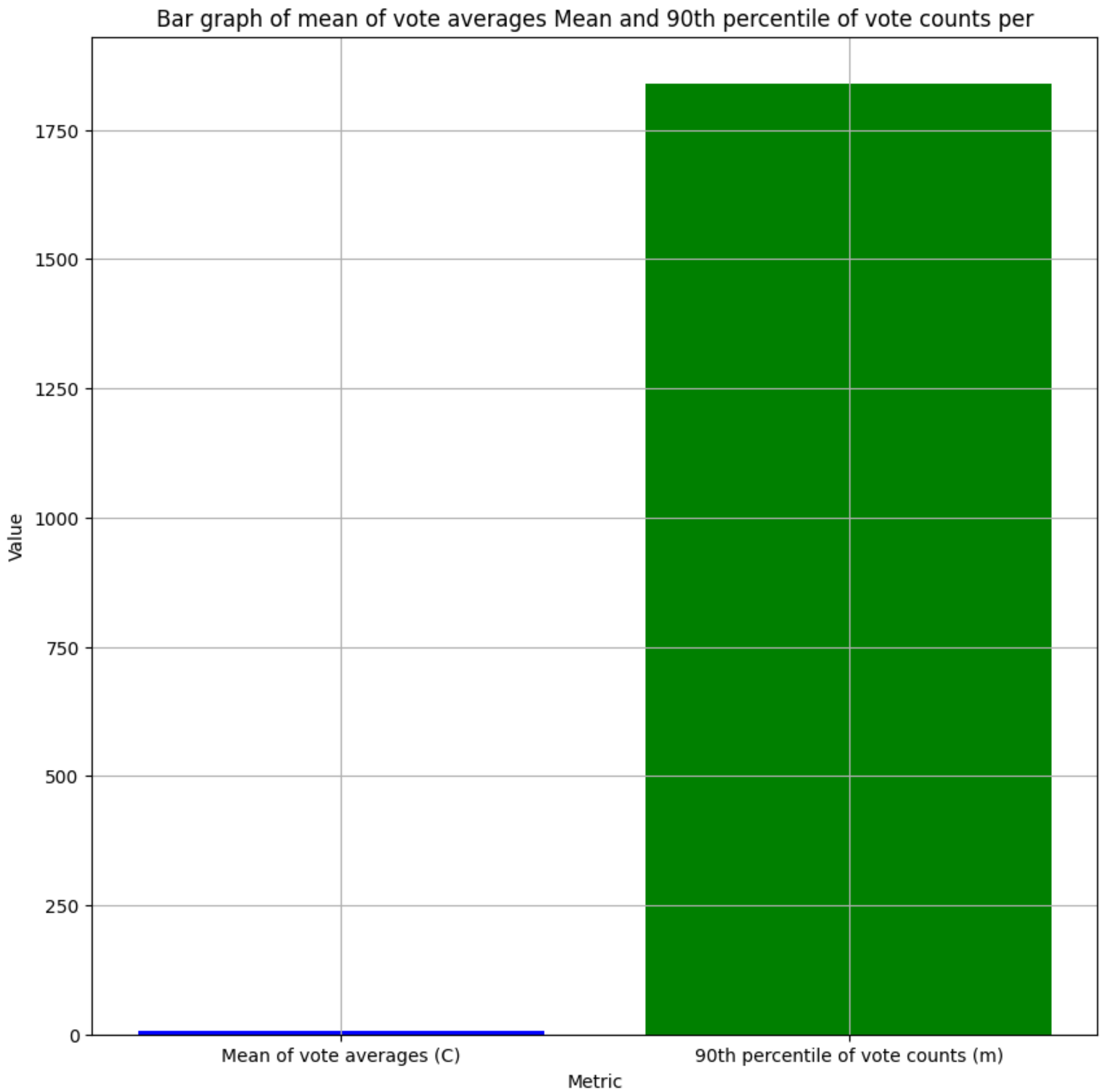
```
                                                cast  \
0  [{"cast_id": 242, "character": "Jake Sully", "…
1  [{"cast_id": 4, "character": "Captain Jack Spa…
2  [{"cast_id": 1, "character": "James Bond", "cr…
3  [{"cast_id": 2, "character": "Bruce Wayne / Ba…
4  [{"cast_id": 5, "character": "John Carter", "c…

                                                crew
0  [{"credit_id": "52fe48009251416c750aca23", "de…
1  [{"credit_id": "52fe4232c3a36847f800b579", "de…
2  [{"credit_id": "54805967c3a36829b5002c41", "de…
3  [{"credit_id": "52fe4781c3a36847f81398c3", "de…
4  [{"credit_id": "52fe479ac3a36847f813eaa3", "de…

[5 rows x 23 columns]
```

1.Data Visualization and Checking The Central Tendancy

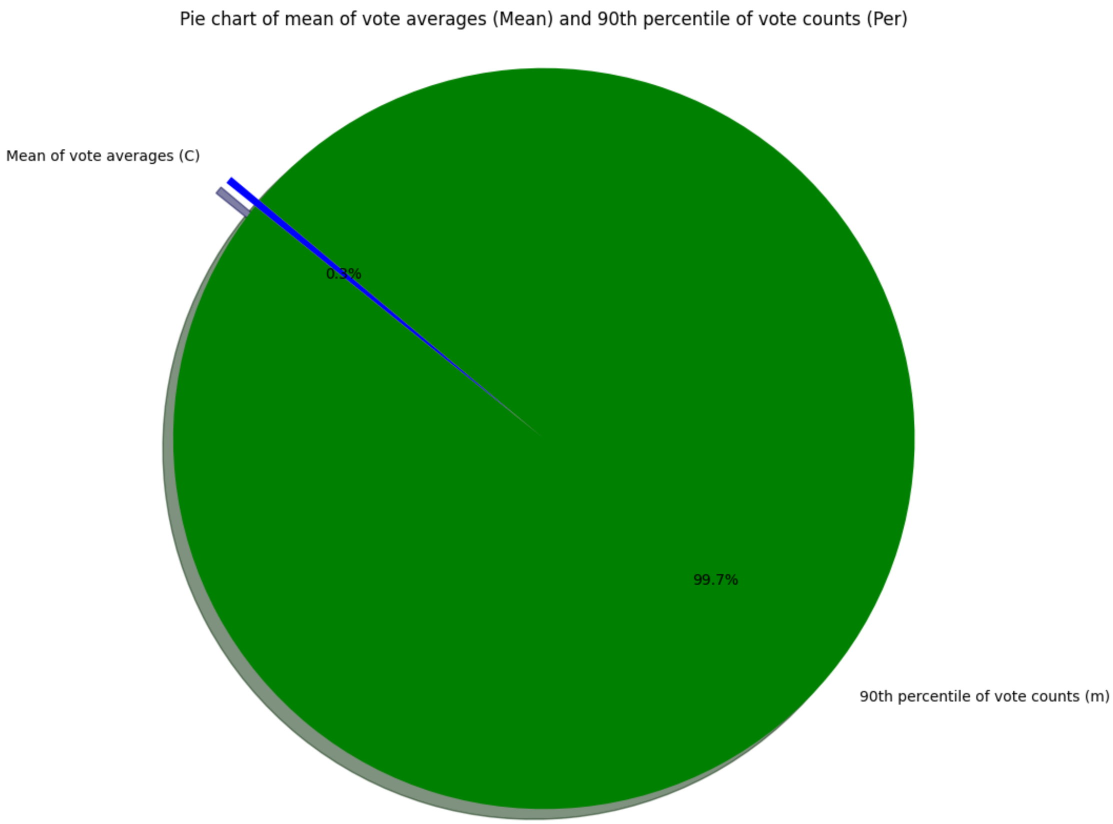
[98]:
```
#check the  Mean and 90th percentile of the Movie dataset
Mean= sub_set2['vote_average'].mean()
print(Mean)
per= sub_set2['vote_count'].quantile(0.9)
print(per)
```

```
6.092171559442016
1838.4000000000015
```

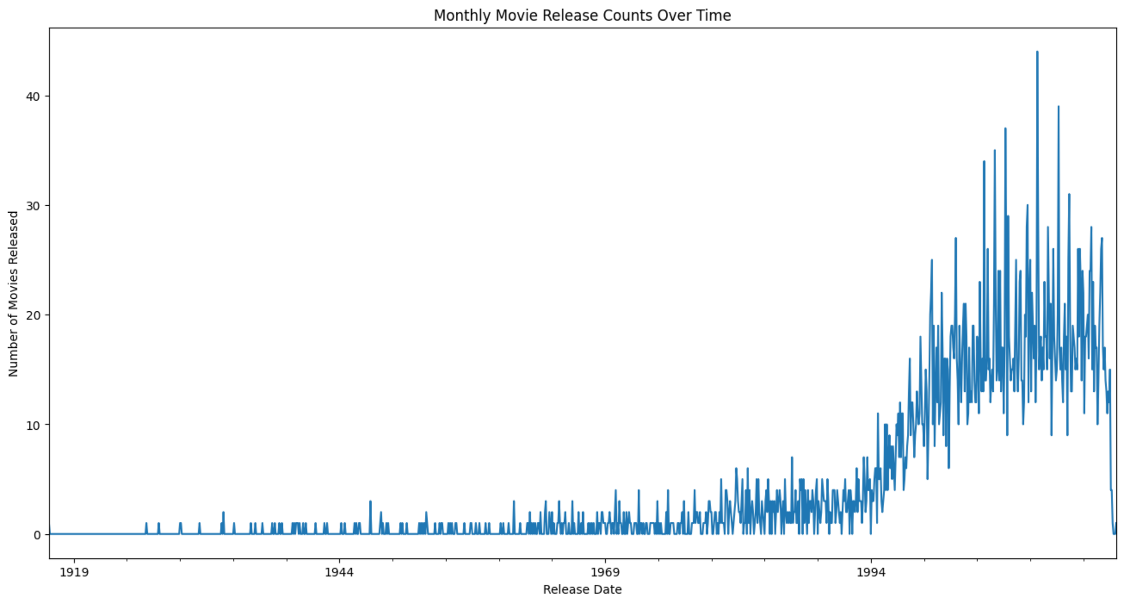
[99]:
```
# Creating a bar graph
plt.figure(figsize=(10, 10))
plt.bar(['Mean of vote averages (C)', '90th percentile of vote counts (m)'],
 ↪[Mean, per], color=['b', 'g'])
plt.xlabel('Metric')
plt.ylabel('Value')
plt.title('Bar graph of mean of vote averages Mean and 90th percentile of vote
 ↪counts per')
plt.grid(True)
plt.show()
```

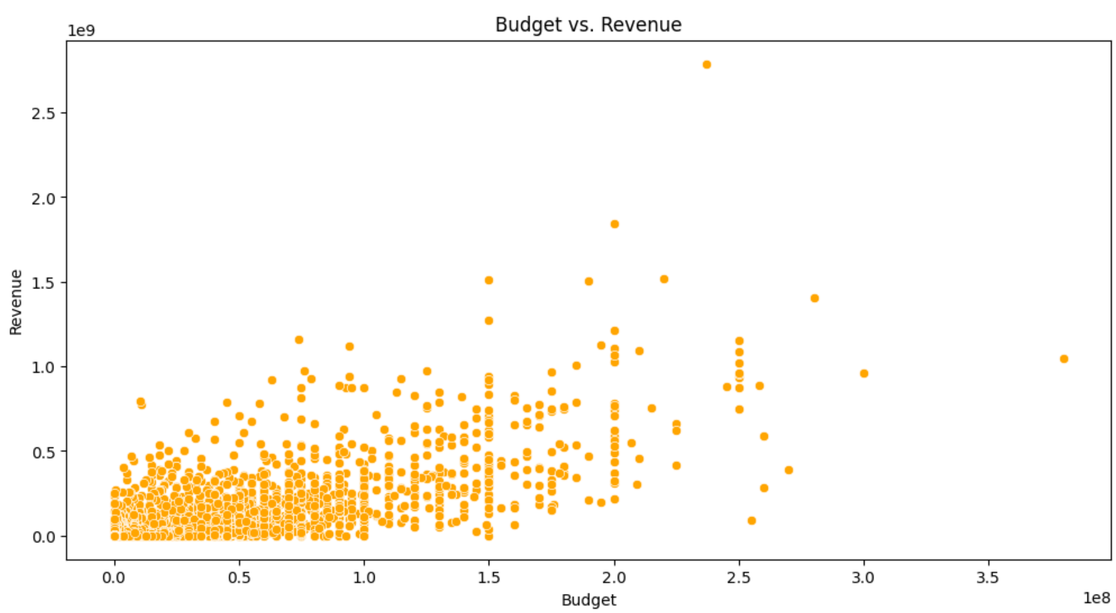


[100]:
```
# Create a pie chart
labels = ['Mean of vote averages (C)', '90th percentile of vote counts (m)']
sizes = [Mean, per]
colors = ['b', 'g']
explode = (0.1, 0)
plt.figure(figsize=(10, 10))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.
↪1f%%', shadow=True, startangle=140)
plt.title('Pie chart of mean of vote averages (Mean) and 90th percentile of␣
↪vote counts (Per)')
plt.axis('equal')
plt.show()
```



[167]:
```
# Release Date Analysis
sub_set2['release_date'] = pd.to_datetime(sub_set2['release_date'])
monthly_movie_counts = sub_set2.resample('M', on='release_date').size()
plt.figure(figsize=(16, 8))
monthly_movie_counts.plot()
plt.title('Monthly Movie Release Counts Over Time')
plt.xlabel('Release Date')
plt.ylabel('Number of Movies Released')
plt.show()

# Budget and Revenue Analysis
plt.figure(figsize=(12, 6))
sns.scatterplot(x='budget', y='revenue', data=sub_set2, color='orange')
plt.title('Budget vs. Revenue')
plt.xlabel('Budget')
plt.ylabel('Revenue')
plt.show()
```





2.Demographic Filtering

[101]:
```
# getting  the shape of  sub movie dataset that is greater than or equal  Mean
q_movies = sub_set2.copy().loc[sub_set2['vote_count'] >= Mean]
q_movies.shape
```

[101]: (4492, 23)

[102]:
```python
# Defining a function for weighted rating based on IMDB formula
def weighted_rating(x, m=per, C=Mean):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
```

[103]:
```python
# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

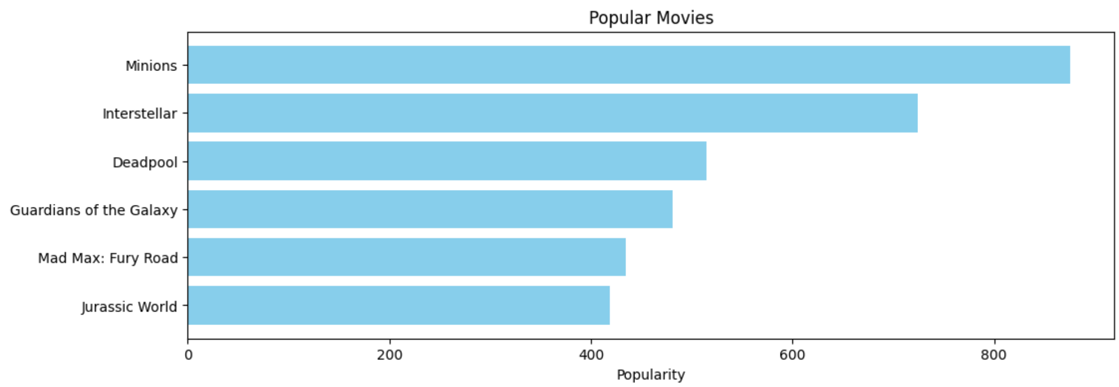
[104]:
```python
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

[104]:
```
                                           title  vote_count  vote_average  \
1881                    The Shawshank Redemption        8205           8.5
662                                   Fight Club        9413           8.3
65                               The Dark Knight       12002           8.2
3232                                Pulp Fiction        8428           8.3
96                                     Inception       13752           8.1
3337                               The Godfather        5893           8.4
95                                  Interstellar       10867           8.1
809                                 Forrest Gump        7927           8.2
329   The Lord of the Rings: The Return of the King        8064           8.1
1990                     The Empire Strikes Back        5879           8.2

         score
1881  8.059258
662   7.939256
65    7.920020
3232  7.904645
96    7.863239
3337  7.851236
95    7.809479
809   7.803188
329   7.727243
1990  7.697884
```

[105]:
```python
# Creating a horizontal bar plot to visualize popular movies
pop= sub_set2.sort_values('popularity', ascending=False)
plt.figure(figsize=(12,4))

plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
        color='skyblue')
plt.gca().invert_yaxis()
```

```
plt.xlabel("Popularity")
plt.title("Popular Movies")
```

[105]: Text(0.5, 1.0, 'Popular Movies')



Content Based Filtering

A recommendation method called content-based filtering makes recommendations to a consumer based on the qualities of products they have previously liked. Content-based filtering algorithms examine attributes that users have found enjoyable in movies, including storyline keywords, actors, directors, and genre, in order to spot trends and preferences when it comes to movie suggestion. The algorithm suggests comparable films that the viewer would probably like based on these trends.

The content-based filtering algorithm, for example, will give recommendations for more comedic films precedence if the user has a history of watching and rating comedies well. This is because the system recognizes the user's apparent liking for humor and lighter amusement.

In addition, the algorithm will recommend movies with actors or directors that the user has indicated they enjoy watching.

Personalized movie suggestions and individualised tastes may be achieved with the use of content-based filtering. The technology can efficiently direct users toward films that match their interests by examining user preferences and seeing trends in their previous selections.

[106]:
```
# Display the overview of the first few movies
sub_set2['overview'].head()
```

[106]:
```
0    In the 22nd century, a paraplegic Marine is di…
1    Captain Barbossa, long believed to be dead, ha…
2    A cryptic message from Bond's past sends him o…
3    Following the death of District Attorney Harve…
4    John Carter is a war-weary, former military ca…
Name: overview, dtype: object
```

[107]:
```
# Text Vectorization
tfidf = TfidfVectorizer(stop_words = 'english')

sub_set2['overview'] = sub_set2['overview'].fillna('')

tfidf_matrix = tfidf.fit_transform(sub_set2['overview'])

tfidf_matrix.shape
```

[107]: (4803, 20978)

Text Vectorization

In machine learning, text vectorization is the process of converting text input into numerical vectors. This is significant since machine learning techniques can only handle numerical data. These two broad categories of text vectorization techniques are:

Techniques that rely on counts: These techniques simply count how many times each word appears in a document. Two techniques that may be applied for this are TF-IDF and Bag-of-words (BoW).

Word embedding techniques include: Rather than focusing only on word counts, these approaches aim to capture the meaning of words and their relationships with one another. There are two ways to accomplish this: Word2Vec and GloVe.

Machine learning applications such as sentiment analysis, topic modeling, and natural language processing (NLP) rely on it.

[108]:
```
# Import TfidfVectorizer for text vectorization
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

[109]:
```
#Construct a reverse map of indices and movie titles
indices = pd.Series(sub_set2.index, index = sub_set2['title']).drop_duplicates()
```

[110]:
```
# Function that takes in movie title as input and outputs most similar movies

def get_recommendations(title, cosine_sim = cosine_sim):

    idx = indices[title]

    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = sorted(sim_scores, key = lambda x:x[1], reverse=True)

    sim_scores = sim_scores[1:11]

    movie_indices = [i[0] for i in sim_scores]

    return sub_set2['title'].iloc[movie_indices]
```

[111]:
```python
# Getting Recommendation
get_recommendations('Stolen')
```

[111]:
```
1868           Cradle 2 the Grave
3905                  Family Plot
3882                        Feast
2400                   The Prince
781                      Inkheart
956     Resident Evil: Apocalypse
2843                    Philomena
3606                    No Escape
1577            Without a Paddle
4513                        Benji
Name: title, dtype: object
```

[112]:
```python
get_recommendations('Plastic')
```

[112]:
```
2923                          St. Trinian's
4268    Lock, Stock and Two Smoking Barrels
2027                               I Am Sam
16                             The Avengers
2212                               Triple 9
1339                            Blue Streak
4124                      This Thing of Ours
39                             TRON: Legacy
4391                        The Perfect Host
3705                         Moms' Night Out
Name: title, dtype: object
```

Credits, Genres and Keywords Based Recommender

[113]:
```python
# Parse the stringified features into their corresponding python objects
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    sub_set2[feature] = sub_set2[feature].apply(literal_eval)
```

[114]:
```python
# Define functions to extract directors  from features
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```

[115]:
```python
# Define functions to get list of names from features
```

```
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]

        if len(names) > 3:
            names = names[:3]
        return names


    return []
```

[116]:
```
sub_set2['director'] = sub_set2['crew'].apply(get_director)

features = ['cast', 'keywords', 'genres']
for feature in features:
    sub_set2[feature] = sub_set2[feature].apply(get_list)
```

[117]:
```
# Print the new features of the first 3 films
sub_set2[['title', 'cast', 'director', 'keywords', 'genres']].head(3)
```

[117]:
```
                                      title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre

                                               cast        director  \
0  [Sam Worthington, Zoe Saldana, Sigourney Weaver]   James Cameron
1     [Johnny Depp, Orlando Bloom, Keira Knightley]  Gore Verbinski
2      [Daniel Craig, Christoph Waltz, Léa Seydoux]      Sam Mendes

                              keywords                          genres
0   [culture clash, future, space war]  [Action, Adventure, Fantasy]
1   [ocean, drug abuse, exotic island]  [Adventure, Fantasy, Action]
2  [spy, based on novel, secret agent]    [Action, Adventure, Crime]
```

[118]:
```
# Function to convert all strings to lower case and strip names of spaces
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ","")) for i in x]

    else:

        if isinstance(x , str):
            return str.lower(x.replace(" ",""))
        else:
            return ''
```

[119]:
```
# Apply clean_data function to your features.
features = ['cast', 'keywords', 'director', 'genres']

for feature in features:
    sub_set2[feature] =sub_set2[feature].apply(clean_data)
```

[120]:
```
def create_soup(x):
    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' +␣
  ↪x['director'] + ' ' + ' '.join(x['genres'])
sub_set2['soup'] = sub_set2.apply(create_soup, axis=1)
```

[121]:
```
# Import CountVectorizer and create the count matrix
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(sub_set2['soup'])
```

[122]:
```
# Compute the Cosine Similarity matrix based on the count_matrix
cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

[123]:
```
# Reset index of our main DataFrame and construct reverse mapping as before
sub_set2 = sub_set2.reset_index()
indices = pd.Series(sub_set2.index, index=sub_set2['title'])
```

[124]:
```
# Prediction Corner
Movie=input("Enter Movie Name to get Other Recommendations:")
get_recommendations(Movie, cosine_sim2)
```

```
Enter Movie Name to get Other Recommendations:Plastic
```

[124]:
```
4247       Me You and Five Bucks
4401         The Helix… Loaded
4638    Amidst the Devil's Wings
1978             Ready to Rumble
2140            Paint Your Wagon
2485                 The Cookout
2650          All The Queen's Men
248             Mr. & Mrs. Smith
685              Blades of Glory
807                  The Pacifier
Name: title, dtype: object
```

Collaborative Filtering

[125]:
```
data = Dataset.load_from_df(sub_set3[['userId', 'movieId', 'rating']], reader)
```

[126]:
```
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'])
```

[126]:
```
{'test_rmse': array([0.90207838, 0.89354729, 0.89633033, 0.88626533,
0.90044671]),
 'test_mae': array([0.69349095, 0.68779844, 0.69117969, 0.68448903,
0.69213631]),
 'fit_time': (1.295379877090454,
  1.2796998023986816,
  1.2026240825653076,
  1.2782962322235107,
  1.3215982913970947),
 'test_time': (0.12032556533813477,
  0.14565587043762207,
  0.29384827613830566,
  0.10508418083190918,
  0.14219141006469727)}
```

[127]:
```
trainset = data.build_full_trainset()
svd.fit(trainset)
```

[127]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7d6040a7cfa0>
```

[182]:
```
sub_set3[sub_set3['userId']==2]
```

[182]:

	userId	movieId	rating	timestamp
20	2	10	4.0	835355493
21	2	17	5.0	835355681
22	2	39	5.0	835355604
23	2	47	4.0	835355552
24	2	50	4.0	835355586
..	…	…	…	…
91	2	592	5.0	835355395
92	2	593	3.0	835355511
93	2	616	3.0	835355932
94	2	661	4.0	835356141
95	2	720	4.0	835355978

[76 rows x 4 columns]

[142]:
```
svd.predict(1, 302, 3)
```

[142]:
```
Prediction(uid=1, iid=302, r_ui=3, est=2.8412843890172788,
details={'was_impossible': False})
```

[189]:
```
from surprise import Dataset
from surprise.model_selection import train_test_split

# Here we load the dataset
reader = Reader()
```

```python
data = Dataset.load_from_df(sub_set3[['userId', 'movieId', 'rating']], reader)

# Split the data into training and testing sets
trainset, testset = train_test_split(data, test_size=0.2, random_state=42)
```

[191]:

```python
# Performing A/B Test
# Comparing SVD vs KNNBasic
a1 = svd
a2 = KNNBasic()

# Function for A/B Test
def ab_test(algorithm1, algorithm2, trainset, testset):
    random.seed(42)  # Set seed for reproducibility

    # Training SVD on the trainset
    a1.fit(trainset)

    # Training KNN on the trainset
    a2.fit(trainset)

    # Evaluating both algorithms
    predict1 = a1.test(testset)
    predict2 = a2.test(testset)

    # Comparing both using RMSE and MAE
    rmse1 = accuracy.rmse(predict1)
    rmse2 = accuracy.rmse(predict2)
    mae1 = accuracy.mae(predict1)
    mae2 = accuracy.mae(predict2)

    # Print the results
    print(f'RMSE for SVD: {rmse1}')
    print(f'RMSE for KNNBasic: {rmse2}')
    print(f'MAE for SVD: {mae1}')
    print(f'MAE for KNNBasic: {mae2}')

    return rmse1, rmse2, mae1, mae2

# Running A/B test
rmse_svd, rmse_knn, mae_svd, mae_knn = ab_test(a1, a2, trainset, testset)

# Compare the results and print which is better
if rmse_svd < rmse_knn:
    print('SVD is better in RMSE.')
else:
    print('KNNBasic is better in RMSE.')
```

```python
if mae_svd < mae_knn:
    print('SVD is better in MAE.')
else:
    print('KNNBasic is better in MAE.')
```

```
Computing the msd similarity matrix…
Done computing similarity matrix.
RMSE: 0.9007
RMSE: 0.9663
MAE:  0.6946
MAE:  0.7452
RMSE for SVD: 0.9007034672888036
RMSE for KNNBasic: 0.9662515187787728
MAE for SVD: 0.694575896676799
MAE for KNNBasic: 0.7451601861211438
SVD is better in RMSE.
SVD is better in MAE.
```