Vivek Poojari
2019130050
TE COMPS
AIML Lab

Experiment 5

Aim: To train and test machine learning model using K-Means Algorithm.

Theory:

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.

The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbor classifier, a popular supervised machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

Code:

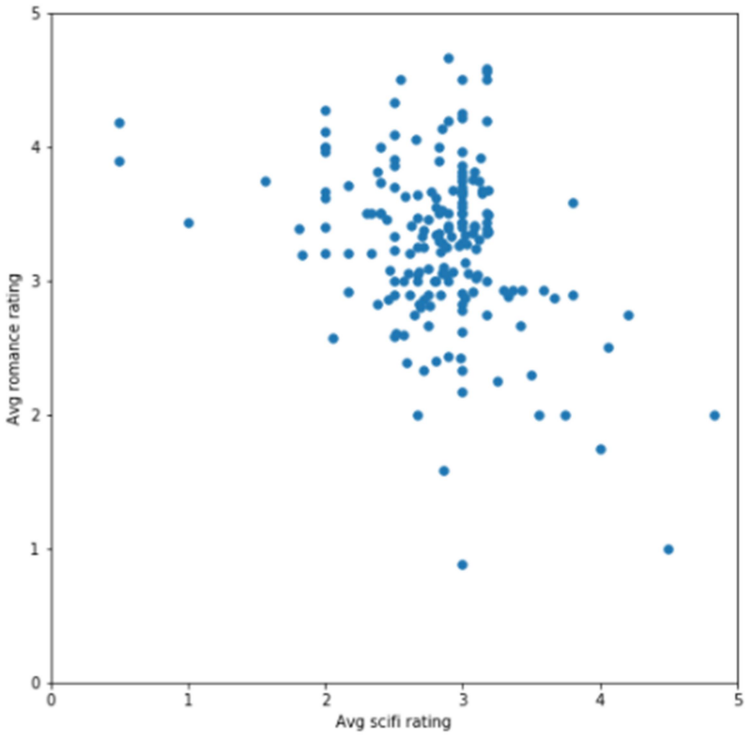
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.sparse import csr_matrix
import helper

# Import the Movies dataset
movies = pd.read_csv('ml-latest-small/movies.csv')
movies.head()

# Import the ratings dataset
ratings = pd.read_csv('ml-latest-small/ratings.csv')
ratings.head()

print('The dataset contains: ', len(ratings), ' ratings of ', len(movies), ' movies.')

# Calculate the average rating of romance and scifi movies

genre_ratings = helper.get_genre_ratings(ratings, movies, ['Romance', 'Sci-Fi'], ['avg_r
genre_ratings.head()

biased_dataset = helper.bias_genre_rating_dataset(genre_ratings, 3.2, 2.5)

print( "Number of records: ", len(biased_dataset))
```

```
print( "Number of records: ", len(biased_dataset))
biased_dataset.head()

%matplotlib inline

helper.draw_scatterplot(biased_dataset['avg_scifi_rating'],'Avg scifi rating', biased_da

# Let's turn our dataset into a list
X = biased_dataset[['avg_scifi_rating','avg_romance_rating']].values

# TODO: Import KMeans
from sklearn.cluster import KMeans



# TODO: Create an instance of KMeans to find two clusters
kmeans_1 = KMeans(n_clusters=2, random_state=0)

# TODO: use fit_predict to cluster the dataset
predictions = kmeans_1.fit_predict(X)

# Plot
helper.draw_clusters(biased_dataset, predictions)
```

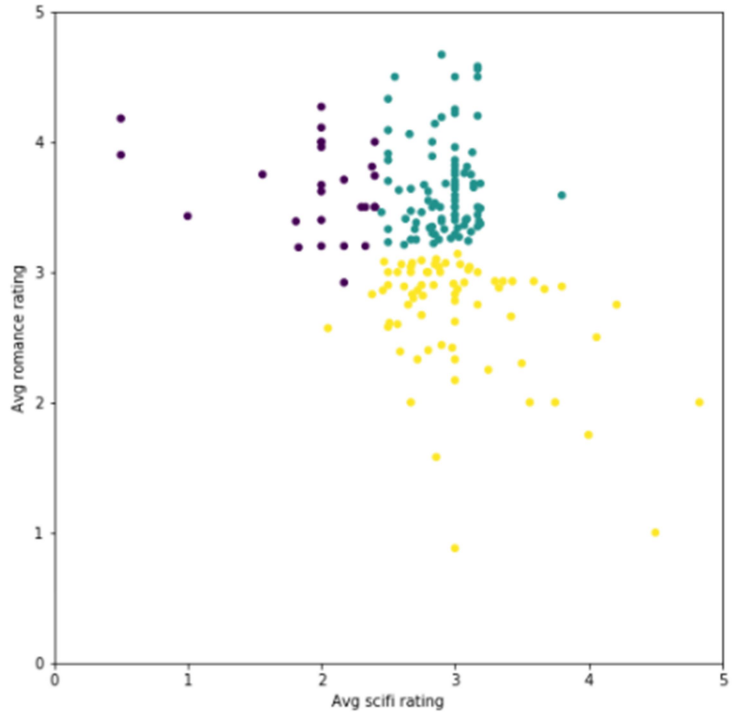
```
# TODO: Create an instance of KMeans to find three clusters
kmeans_2 = KMeans(n_clusters=3, random_state=1)

# TODO: use fit_predict to cluster the dataset
predictions_2 = kmeans_2.fit_predict(X)

# Plot
helper.draw_clusters(biased_dataset, predictions_2)




# TODO: Create an instance of KMeans to find four clusters
kmeans_3 = KMeans(n_clusters=4, random_state=3)

# TODO: use fit_predict to cluster the dataset
predictions_3 = kmeans_3.fit_predict(X)

# Plot
helper.draw_clusters(biased_dataset, predictions_3)
```

Output:

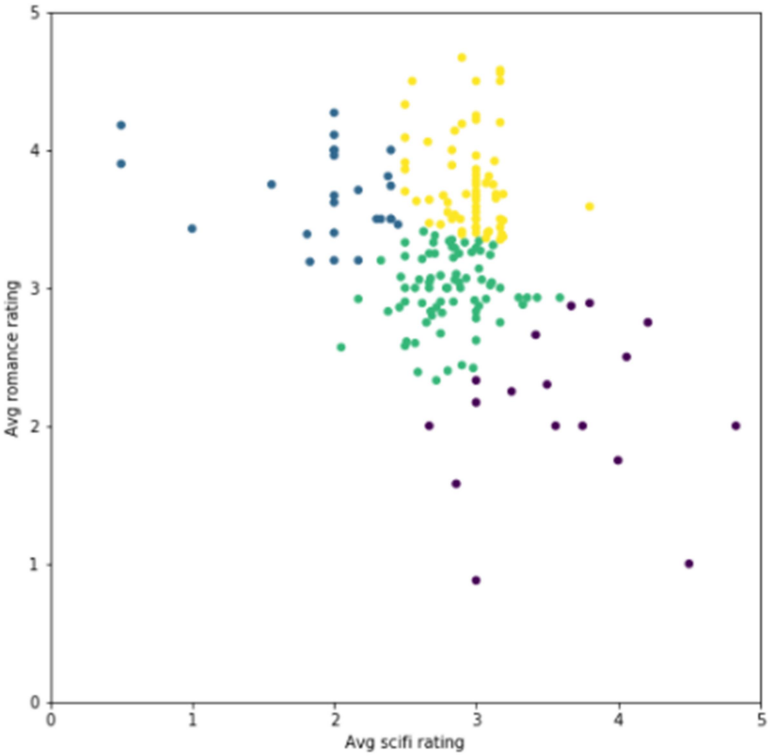
Plot 1: Avg Romance rating vs Average scifi rating



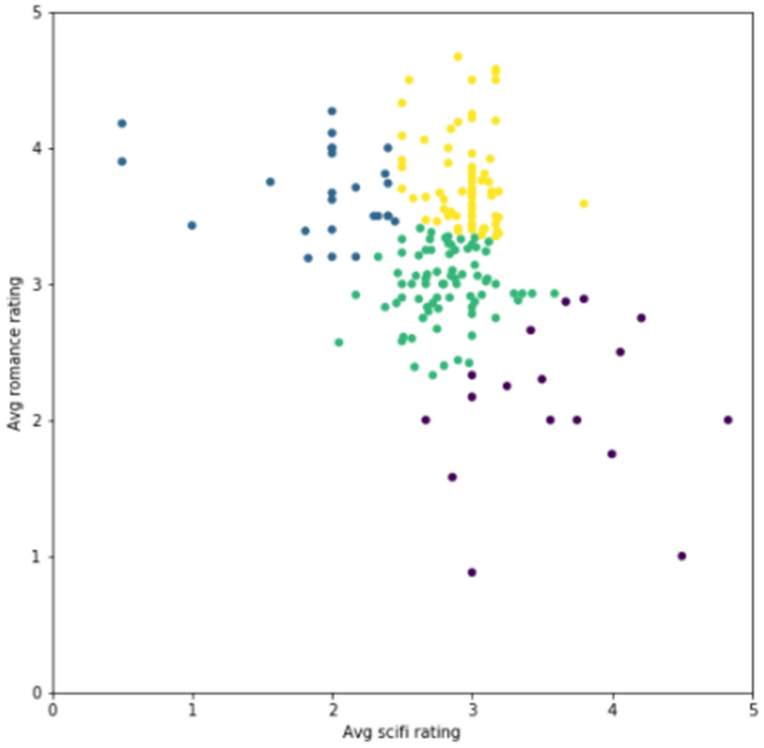
Plot 2: K=2 for Avg scifi rating against Avg romance rating



Plot 3: K=3 for Avg scifi rating against Avg romance rating



Plot 4: K=4 for Avg scifi rating against Avg romance rating



Conclusion:

I implemented the K-Means algorithm in this experiment and run the algorithm on a dataset called Movielens - user rating dataset. The purpose was to explore the similarities and differences in people's tastes in movies based on how they rate different movies. Using k-means algorithm graphs were plotted for 'k' values 2,3,4. We can see that the more clusters we break our dataset down into, the more similar the tastes of the population of each cluster to each other.

So I was able to grasp the basics of the K-Means method. Each cluster has its own centroid, called as the centroid-based approach. Reducing the sum of distances between data points and the clusters that they belong to, is the main goal of this technique.