**MOVIE RECOMMENDATION SYSTEM ON BIG DATA**

Dr. T. ADILAKSHMI (HOD - CSE),

Ms. S. SREELAKSHMI (ASSISTANT PROFESSOR (SR. SCALE),

L. RAHUL, T. ANUVIK

**Abstract**

Recommender systems are being used extensively in the modern digital age as mammoth of user data is available in the contemporary private databases. Handling large amounts of data has become stressful due to proliferation of user data available on the web. The existing recommender systems are built using algorithms like K-means, Collaborative filtering etc. Recommendation on Big Datasets where content is not given needs to be replaced and work in parallel with tag-genome data.

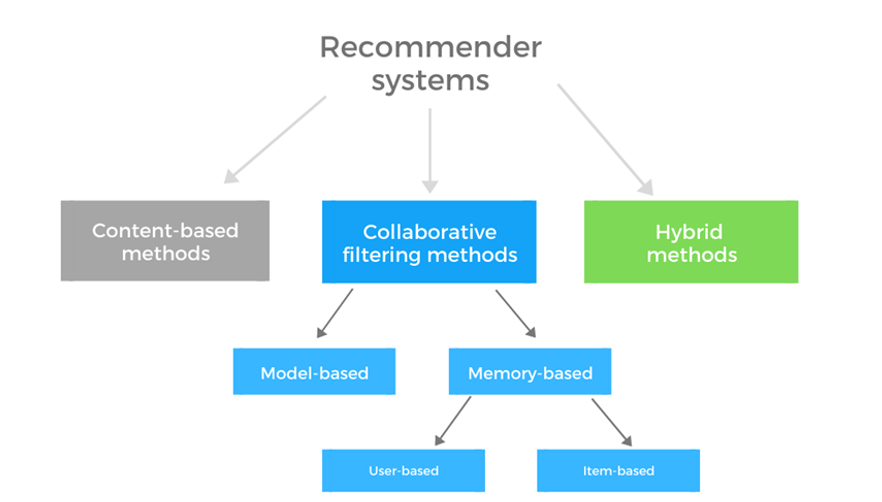
In this study, we used CURE Algorithm for clustering the data. Pre-eminent clustering algorithms are either bent towards Spherical shape clusters with close sizes or sensitive with respect to others. Here, we are introducing and implementing a new methodology which is more suitable to handle the outliers which even suggests clusters of non-spherical shape of heterogenous sizes. CURE Algorithm helps in introducing another method which extrapolating every cluster by fixing the number of points which are obtained from random points in the cluster which with the help of precise fraction are directed towards the cluster centre. For managing voluminous databases, CURE implements a mix of partitioning and random sampling. Firstly, a partition set is devised from the random sample and then it is partially clustered.

Recommendation system helps in prediction of the rating of the item or to suggest item to the user.

**Goals of Recommendation System:**

**1.User-Experience Customization:** Enhances the hands-on experience for the user by analysing the user’s behaviour and preferences.

**2.Optimum Item Search:** Splits the items based on the parameters.

****

**Fig 1:** Methods of Recommender Systems

**1.1 Content-Based Recommendation System:**

In this methodology, the system suggests user’s preferred item based on their selection criteria by using similarity. It suggests items based on their existing functionalities. It recognizes the closeness among the items on the basis of content.

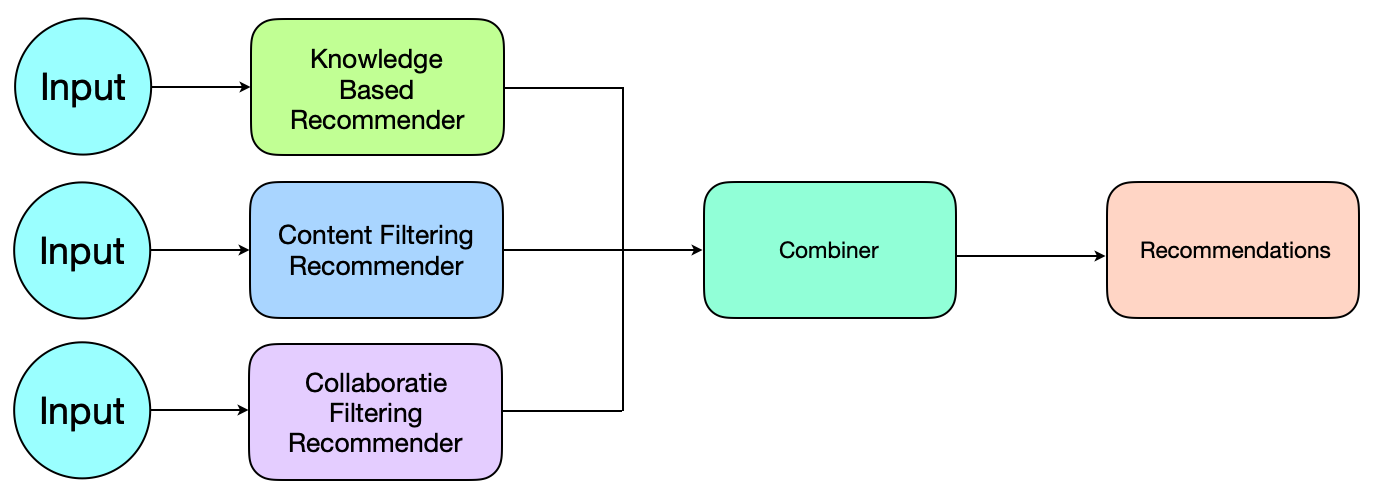
**1.2 Collaborative Based Recommendation System:**

This methodology suggests items based on collaboration of users which are luminous methods used in market.

**1.3 Hybrid Recommendation System:**

Hybrid Recommendation System is an amalgamation of Content and Collaborative based recommendation systems.

We used Hybrid recommendation system in our project to recommend movies to user.



**Fig 2:** Illustrates the working of hybrid recommendation system. It comprises of both collaborative and content-based filtering.

Here, we used the dataset which comprises of 1128 Tag Relevance values (Movie belongingness to a particular tag) and 228 movies. Tag Genome records indicate how strongly a tag is related to item ranging from 0 to 1, where ‘0’ is the weakest and ‘1’ is strongest. Here, we used feature reduction for reducing 1128 tags to 300 clustered tags. The clustering process here is done using Pearson Correlation Metric using Agglomerative clustering.

**1.4 Parallelization:**

Spark is known for its parallel processing, which means a dataframe or a resilient distributed dataset (RDD) is being distributed across the worker nodes to gain maximum performance while processing. Using Databricks, we implement multi-threading is relatively quick to set up compared with other optimization methods.

The improvement could be unlimited if we have a large enough cluster and plenty of jobs to run parallelly. The purpose of using multi-threading is not only to save time, but also to fully utilize the clusters’ compute power to save cost by finishing the same amount of jobs within less time, or within the same amount of time on a smaller cluster, which gives us more options to manage the end-to-end pipeline.

**2. Method**

**2.1 CURE**

The Clustering Algorithm used is Clustering Using Representatives (CURE). The pivotal reason for opting this Algorithm is its compatibility with Big Data. The Crux of this algorithm is to represent a particular cluster in terms for few Representative points, there by scaling the large dataset into few representative points helps in handing gigantic amount of data.

Also CURE incorporates a compression factor that reduces the cluster size to some imaginary level and detects anomalies, thereby providing effective clusters. This Algorithm can be parallelized using distributed environment and process the application for Big Data.

A picture containing text

Description automatically generated Text

Description automatically generated

**Fig 3(a).** Working of the Cure Clustering Algorithm **Fig 3(b).** Merging Phase of the Algorithm

**2.2 Data Pre-processing**

The Core dataset which is used is the **MovieLens** Tag genome Dataset. The Dataset we used consists of 1128 of Tags with their Corresponding movies Tag Relevance scores. Since the use of ratings might make the recommendation system obsolete, we used factual data to identify the category of the movie.

These 1128 tags consist of the mixture of components cumulatively calculated to give a single Tag Id. This Tag Id indicates it’s on categorical entity for a corresponding movie. The relevance scores lie between the range of **0 to 1 (0 least relevant, 1 most relevant)**.

**2.2.1 Feature Reduction**

The starting step is the feature reduction, we reduced the 1128 tag features to few sets of clustered tags.

1. 300 Clustered Tags
2. 500 Clustered Tags
3. 700 Clustered Tags

The clustering process here is done using **Pearson Correlation** Metric using **Agglomerative Clustering.**

Chart, histogram

Description automatically generated

**Fig 4:** A Dendrogram showing the relationship between 1128 tags(Tag Relevance scores).

The Labels after the clustering will be grouped based on mean of the tag relevance values of the clusters.

**2.2.2 Experimentation with 700 Clustered Tags**

Firstly, we considered to restrict the 1128 Tags to 700 clusters. We reduced the 1128 Tags to 700 Clustered Tags using Pearson Correlation.The 1128 Tags are clustered into 700 Clustered Tags.For 700 Tags and 228 Movies We applied CURE to cluster the movies.

**CURE PARAMETERS**

* Representative points = 5
* Alpha = 0.2
* Optimal Number of clusters = 8

Chart, scatter chart

Description automatically generated

**Fig 5:** Graph Showing Relation between the Within Sum of Clusters Scores (WSS) on

X - Axis and number of Clusters on Y – Axis for 700 Clustered Tags.

**Silhouette score Analysis**

The silhouette scores deduced for 700 features gave negative results, indicate that the movies are assigned to wrong clusters

Chart, scatter chart

Description automatically generated

**Fig 6:** Silhouette score analysis for 700 clustered Tags, Graph shows the **silhouette scores** on Y- Axis and Values of **K** (Number of Clusters) on X- Axis for 700 clustered Tags.

**2.2.2 Experimentation with 500 Clustered Tags**

To improve the cluster efficiency, we reduced the features and clustered the 1128 tags into 500 Clustered tags.

The 1128 Tags are clustered into 500 Clustered Tags

For 500 Tags and 228 Movies We applied CURE to cluster the movies.

**CURE PARAMETERS**

* Representative points = 5
* Alpha = 0.2
* Optimal Number of clusters = 6

Chart, scatter chart

Description automatically generated

**Fig 7:** Graph Showing Relation between the Within Sum of Clusters Scores (WSS) on

X - Axis and number of Clusters on Y – Axis for 500 Clustered Tags.

**Silhouette score Analysis**

The silhouette scores obtained for 500 features gave negative results, meaning movies are assigned to wrong clusters

Text

Description automatically generated

**Fig 8:** Output Console showing the results of the silhouette scores of movies clusters obtained from 500 Clustered Tags.

**2.2.3 Experimentation with 300 Clustered Tags**

The features were further reduced to 300 Clustered Tags

1128 tags into 300 Clustered tags.

For 300 Tags and 228 Movies We applied CURE to cluster the movies.

**CURE PARAMETERS**

* Representative points = 5
* Alpha = 0.2
* Optimal Number of clusters = 8

Chart, scatter chart

Description automatically generated

**Fig 9:** Graph Showing Relation between the Within Sum of Clusters Scores (WSS) on

X - Axis and number of Clusters on Y – Axis for 300 Clustered Tags.

**Silhouette score Analysis**

Chart, scatter chart

Description automatically generated

**Fig 10:** Silhouette score analysis for 300 clustered Tags, Graph showing the **silhouette scores** on Y- Axis and Values of **K** (Number of Clusters) on X- Axis for 300 clustered Tags.

The silhouette scores of the movie clusters for feature reduction to 300 Clusters were positive, and hence we reduced the tag scores to 300 Clustered Tags.

**2.3 Scaling up the process to 1000 user’s dataset.**

**2.3.1 Databricks**

Databricks is a centralized and open-source platform to process all the data. It provides an environment for interaction and scheduling of workloads of data analysis for data analysts, data engineers and data scientists.

**2.3.2 Configurations of Databricks**

The Databricks Runtime version - 10.4 LTS (includes Apache Spark 3.2.1, Scala 2.12)

Driver Type – 15.3 GB Memory, 2 Cores, 1 DBU (Data Brick Unit)

**2.3.3 Spark Data frame**

Firstly, to ensure parallelization the data frame of the movie’s dataset needs to be converted into spark data frame. The spark data frame internally divides it into multiple partitions, where different node are mapped to partition for handling. In this way the parallelization is achieved, parts of the data frame is computed parallelly and the results are merged to give the final output.

|  |  |
| --- | --- |
| Graphical user interface  Description automatically generated | **Text  Description automatically generated** |
| **Fig 11(a).** Figure showing the individual partitions of a spark data frame created using spark session on moviesClusters.csv file | **Fig 11(b).** Number of jobs executed in FAIR Scheduling mode taking an average uptime of 9 minutes. |
| Graphical user interface, text  Description automatically generated with medium confidence | |
| **Fig 11(c).** Spark Context created to run the application in Databricks shell | |

**2.3.4 1000 Users Dataset**

The same algorithm is used for 1000 users as well. Since the data here is large compared to the previous dataset, we are scaling it up using Databricks distributed environment.

The results obtained from previous feature reduction includes **300** Clustered tags and **10381** Movies along with **1000** Users Rating Matrix.

The CURE Algorithm applied to the dataset and corresponding clusters are obtained for all the movies.

A picture containing shape

Description automatically generated

**Fig 12:** Graph Showing Relation between the Within Sum of Clusters Scores (WSS) on Y - Axis and number of Clusters on X – Axis for 300 Clustered Tags.

**2.4 Clustering Users with Similar ratings**

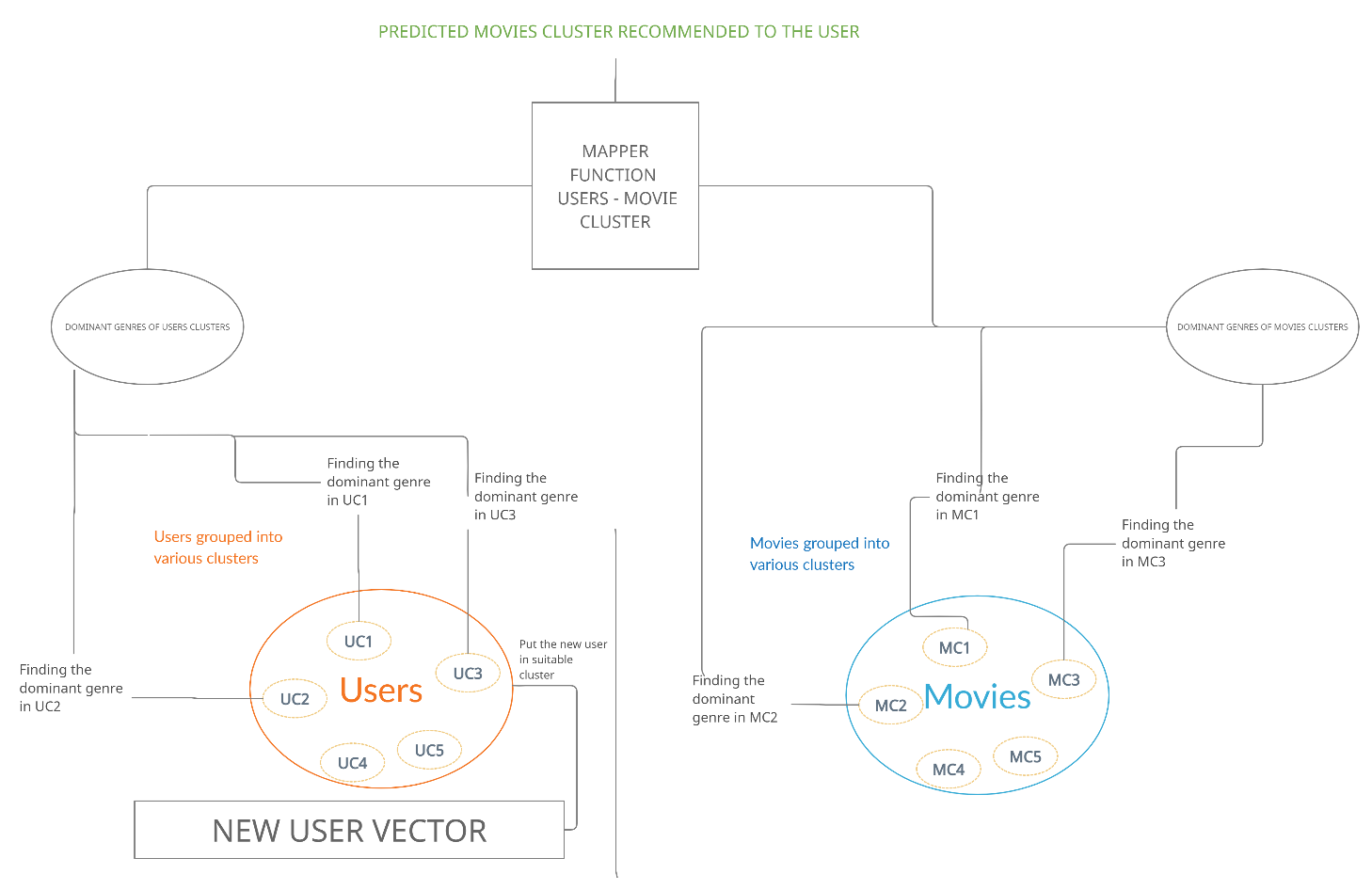
The next part of the algorithm is to cluster the users with similar ratings using the CURE Clustering Algorithm. We found the optimal number of clusters using Elbow Method.Optimal number of clusters obtained are 24.

A picture containing chart

Description automatically generated

**Fig 13:** Graph Showing Relation between the Within Sum of Clusters Scores (WSS) on X - Axis and number of Clusters on Y – Axis for 1000 Users.

**2.5 Hybrid model we proposed: Content Based Filtering of movies and Collaborative filtering of users.**



**Fig 14(a):** Workflow/Algorithm of hybrid recommendation system

**2.5.1 Content Based Filtering on the movies Clusters**

The new model that we proposed uses content-based filtering of movies based on tag genome dataset. All the similar movies are clustered using the CURE Algorithm.

The movie clusters obtained consists of ID’s of the movies belonging to the cluster.

The approach we followed is:

1. Identify the movies present in the corresponding cluster obtained from CURE.
2. Identify the genres of the movies present in the given cluster.
3. Store the key value pairs of the cluster ID’s and movie ID’s and identify the genres present in the cluster.
4. Find the dominant genre present in the cluster.
5. Repeat step 1 to 4 for rest of the clusters.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated with medium confidence |
| **Fig 14(b):** Output console of the execution showing movie Clusters and genres associated to it. | **Fig 14(c):** Output console of the execution showing movie Clusters favourite genres. |

**2.5.2 Content Based Filtering on the Users clusters**

The user clusters obtained consists of ID’s of the movies belonging to the cluster.

We selected the movies which are favourite to the users i.e. ratings >= 3.

The approach we followed is:

1. Identify the users present in the corresponding cluster obtained from CURE.
2. Identify the genres of the movies favourite to the users present in the cluster.
3. Store the key value pairs of the cluster ID’s and movie ID’s and identify the genres present in the cluster.
4. Find the dominant genre present in the cluster.
5. Repeat step 1 to 4 for rest of the clusters.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |
| **Fig 15(a):** Output console of the execution showing genres associated to favourite movies of the users of the clusters. | **Fig 15(b):** Output console of the execution showing favourite genres of user clusters. |

Once we perform the content-based filtering on both the movies and users, we followed collaborative filtering to recommend movies to users.

Recommending movies to the user done in two phases:

**Phase 1:**

Obtaining the users corresponding cluster and find the favourite genre of the users cluster he grouped into.

**Phase 2:**

Mapping the user to the movie clusters through the favourite genre of the cluster. The movie cluster which matches the favourite genre of both the user and movies will be recommended to the user by sending its predicted cluster.

The predicted cluster contains the movies which might be favourite the current user.

**3. Results**

Calendar

Description automatically generated with medium confidence

**Fig 16:** Spark Jobs executing parallelly in multiple pools

After the previous process we obtain a predicted cluster from the recommendation engine.

All the movies present in the predicted cluster will be the movies which are to be recommended to the user.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |
| **Fig 16(a):** Output console of execution showing recommended movies for user id 987 | **Fig 16(b):** Output console of execution showing recommended movies for user id 153. |

The entire content-based filtering is done using clustering with CURE algorithm. We need to obtain a measure on the reliability of the hybrid recommendation engine we created. We found the **Root Mean Squared Error (RMSE)** of the engine by the following steps.

FINDING THE **RMSE** AND **MAE**

* **Step1:** Split the movies clusters dataset we obtained after clustering the Movie-Tags dataset into desired number of clusters into train and test dataset.
* **Step2:** After the train-test split we found the ratings of the users for the movies present in the test dataset, the actual ratings of the movies is present in **ratings.csv** (1000 Users) file.
* **Step3:** Now we find predicted ratings of the users for the movies present in the test dataset.
* **Step4:** After obtaining the predicted ratings of the user we find the difference between the predicted and actual ratings.
* **Step4:** Now we find the RMSE of each and every rating and then find the average of it.

After following the above steps our model gave an RMSE of **0.98918** and Mean Absolute Error of **0.76129**.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |
| **Fig 17(a):** Output console showing the RMSE value 0.98918. | **Fig 17(b):** Output console showing the MAE value 0.76129. |

1. **Conclusion**

In present generation there is enormous amount of data present, we have variety of recommendation algorithms running specific to the area of interest and domain of the problem.

We can increase the effectiveness of the recommendations by using new variants i.e., hybrid recommendation algorithms and come up with new and efficient algorithm.

The algorithm which we came up with is one such hybrid algorithm to provide proper and reliable personalized recommendations by considering factual data. However, it is inevitable that the data is large and needs to be handled in a faster way. So, the idea of parallelization helped us in reducing the data into various smaller chunks and process them parallelly and utilize the resources effectively. Since we used spark distributed environment this project can be scaled up to real world by using the **AWS S3** Service and process data in real-time.

1. **References**

1. N. Beckmann, H.-P. Kriegei, R. Schneider, and B. Seeger. The R\*-tree: an efficient and robust access method for points and rectangles. In Proc. of ACM SIGMOD, pages 322-331, Atlantic City, NJ, May 1990.

2. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algorithms. The MIT Press, Massachusetts, 1990.

3. Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial database with noise. In I&‘1 Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), Portland, Oregon, August 1996.

4. Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. A database interface for clustering in large spatial databases. In Int’l Conference on Knowledge Discovery in Databases and Data Mining (KDD-95), Montreal, Canada, August 1995.

5. Sudipto Guha, R. Rastogi, and K. Shim. CURE: A clustering algorithm for large databases. Technical report, Bell Laboratories, Murray Hill, 1997.

6. Drineas, Petros; Frieze, Alan M.; Kannan, Ravi; Vempala, Santosh,Vinay,Vishwanathan(2004).

"Clustering large graphs via the singular value decomposition" (PDF). Machine Learning. 56 (1–3): 9-33. doi:10.1023/b:mach.0000033113.59016.96. S2CID 5892850.

7. H. Halpin, V. Robu, and H. Shepherd. "The Complex Dynamics of Collaborative Tagging”,

Proceedings of the 16th international conference on World Wide Web, ACM, USA, 2007, pp. 211-220.

8.  [Peter J. Rousseeuw](https://en.wikipedia.org/wiki/Peter_J._Rousseeuw) (1987). ["Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis"](https://doi.org/10.1016%2F0377-0427%2887%2990125-7). Computational and Applied Mathematics. 20: 53–65. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1016/0377-0427(87)90125-7](https://doi.org/10.1016%2F0377-0427%2887%2990125-7).

9. R.C. de Amorim, C. Hennig (2015). "Recovering the number of clusters in data sets with noise features using feature rescaling factors". Information Sciences. 324: 126–145. [arXiv](https://en.wikipedia.org/wiki/ArXiv_(identifier)):[1602.06989](https://arxiv.org/abs/1602.06989). [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1016/j.ins.2015.06.039](https://doi.org/10.1016%2Fj.ins.2015.06.039).

10. Pavel V. Kolesnichenko; Qianhui Zhang; Changxi Zheng; Michael S. Fuhrer; Jeffrey A. Davis (2021). ["Multidimensional analysis of excitonic spectra of monolayers of tungsten disulphide: toward computer-aided identification of structural and environmental perturbations of 2D materials"](https://iopscience.iop.org/article/10.1088/2632-2153/abd87c). [Machine Learning: Science and Technology](https://en.wikipedia.org/w/index.php?title=Machine_Learning:_Science_and_Technology&action=edit&redlink=1). 2 (2): 025021. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1088/2632-2153/abd87c](https://doi.org/10.1088%2F2632-2153%2Fabd87c).