Assignment Report, CZ4032 Data Analytics and Mining

TMDb Data Mining

Group Member	Matric Number	Contribution
Li Bingzi	U1722793H	Data Exploration, Clustering Analysis
Wu Ziang	U1722432E	Data Preprocessing, Association Rule
Zhang Shengjing	U1722023K	Data Exploration, Clustering Analysis
Zhang Yuehan	U1722863L	Data Exploration, Classification

Abstract

TMDb Data Mining project explores the movie database and attempts to mine data with classification, clustering analysis and association rule techniques.

Some meaningful patterns and relationships among attributes are discovered, such as the association between Drama and Comedy movies, the recommendation of movies based on preference and favorites are achieved as an outcome of data exploration, and the prediction of the success of a movie is also learnt through classification.

Problem Description

Motivation

Data Mining is a non-trivial extraction of implicit, previously unknown, and potentially useful information from data. This motivates to mine the data, in the hope that some meaningful patterns and valuable knowledge can be extracted and inferred out of the data.

Problem Definition

This assignment project explores and analyzes the TMDb movie data on the cast, crew, budget, release date, runtime, revenue, overview, genres, keywords, popularity, vote and other features of several thousand films.

The project aims to discover some meaningful patterns and valuable knowledge such as:

- understand the patterns of attributes and the relations between attributes,
- predict the success of a movie before released,
- and recommend movies based on some criteria.

Related Work

Following are some related work published by others on Kaggle, the online community of data scientists and machine learners:

- Movie Dataset Exploration
  https://www.kaggle.com/epfreed/tidydata-movie-dataset-exploration/report
- Principal Component Analysis with K-Means visuals
  https://www.kaggle.com/arthurtok/principal-component-analysis-with-kmeans-visuals
- Movie Recommendation Systems Based for TMDB
  https://www.kaggle.com/erikbruin/movie-recommendation-systems-based-on-tmdb
- Film Recommendation Engine
  https://www.kaggle.com/fabiendaniel/film-recommendation-engine

Approach

Data Preprocessing

Data preprocessing of the TMDb movie data involves several steps: data integration, data cleaning, feature creation, and finally feature subset selection.

Data Exploration

Methodologies used in data exploration are summary statistics, correlation and visualization.

Association Rule

Apriori algorithm selected is an association rule mining algorithm that optimizes the frequent itemset generation and rule generation based on anti-monotone property.

Classification

Five classifiers, K-Nearest Neighbors, Gaussian Naïve Bayes, Support Vector Machines, Artificial Neural Networks, and Ensemble Classifiers are selected to predict the popularity. To evaluate the classification, Accuracy Score, Confusion Matrix, Precision, and ROC Curve are used as the metrics of the performance.

Clustering Analysis

Three clustering analysis algorithms are chosen, DBSCAN, Agglomerative, and K-Means.

Implementation

Data Preprocessing

Data preprocessing is implemented using mainly NumPy and Pandas packages, with help from some other Python packages and utilities.

Data Exploration

Data is implemented using Pandas for summary statistics and correlation matrix, and the PyPlot from Matlibplot for visualization, with help from Seaborn and some other packages.

Data Mining

Association Rule

Apriori algorithm is implemented by apyori class is imported from apriori.

Classification

Five classification methods and their variations are implemented to classify the popularity of movies. The training and testing data is split on a ratio of 70:30. The classifiers are imported from Scikit-Learn packages.

The label class is popularity and the classifiers predict whether an unseen movie would be popular based on its attributes.

K-Nearest Neighbors Classifier

K-Nearest Neighbors classifier is implemented by KNeighborsClassifier from neighbors:
`KNeighborsClassifier(algorithm='auto', n_neighbors=7)`

- `n_neighbors=7`: the classifier looks for k neighbors vote and decide the label.
- `algorithm='auto'`: the classifier automatically determine the best algorithm.

Naïve Bayes Classifier

Naïve Bayes classifier is implemented by GaussianNB class from sklearn.naïve_bayes:
`GaussianNB(priors=None, var_smoothing=1e-09)`

- `prior=None`: the prior probabilities of classes is adjusted according to the data.
- `var_smoothing=1e-09`: the portion of the largest feature variance is added to data variances for calculation stability.

Support Vector Machines

Support Vector Machines is implemented by NuSVC from svm:
`NuSVC(kernel='linear')`

- `kernel='linear'`: the kernel of the classifier is linear.

Artificial Neural Networks

Artificial Neural Networks is implemented by MLPClassifier from neural_network:

```
MLPClassifier(hidden_layer_sizes=(100),                   max_iter=2000,
                 n_iter_no_change=100, solver='adam')
```

- `hidden_layer_sizes=(100)`: the element of the tuple represent the number of neurons in the corresponding hidden layer.
- `solver='adam'`: adam is a stochastic gradient-based optimizer, which works pretty well on relatively large datasets in terms of both time complexity and validation score.
- `max_iter=2000`: the solver iterates until convergence or for maximum iterations. For 'adam' solver, this determines the number of epochs.
- `n_iter_no_change=100`: when the loss or score is not improving by at least that number of consecutive iterations, a convergence is considered to be reached.

Ensemble Classifier

Ensemble Classifier is implemented by VotingClassifier from ensemble:

```
VotingClassifier(estimators=[('lr',neigh),('rf',gaus),('gnb',mlp)],
                    voting='hard')
VotingClassifier(estimators=[('lr',neigh),('rf',gaus),('gnb',mlp)],
                    voting='soft')
VotingClassifier(estimators=[('lr',neigh),('rf',gaus),('gnb',mlp)],
                 voting='soft',weights=[2,1,1], flatten_transform=True)
```

- Three variations are implemented, Hard, Soft, and Weightage.
- `estimators=[('lr',neigh),('rf',gaus),('gnb',mlp)]`: three classifiers, K-Nearest Neighbors, Naïve Bayes and Artificial Neural Networks, are ensembled.
- `voting='hard'`: the classifier predicts class labels by the majority rule voting.
- `voting='soft'`: the classifier predicts the class label based on the argmax of the sums of the predicted probabilities.
- `weights=[2,1,1]`: the weight class probabilities before averaging. for soft voting

Clustering Analysis

K-Means

K-Means Clustering is implemented by KMeans imported from sklearn.cluster:

```
KMeans(n_clusters=3, max_iter=10000)
```

- **`n_clusters=2`**: 3 clusters is outputted from the clustering. When it comes to some close-packed data set, larger number of clusters can be chosen because smaller set of clusters show the relationships of each point in a much clearer way.
- **`max_iter=10000`**: the maximum number of iterations is set to 10000. In fact, most of the convergence happens in the first few iterations.
- Select the initial centroids is important to the performance of k-means clustering.

Distance Matrix

A self-defined method computes the similarity matrix from the vectorized data. This matrix is passed into DBSCAN and Agglomerative Clustering from sklearn.cluster.

The distance matrix is defined as: the distance between two values of the same feature, categorical values after bucketization and vectorization, is naturally 0 if they are the same and 1 if they are different. This precomputed distance matrix is input for the clustering.

DBSCAN

DBSCAN is implemented by DBSCAN:

```
DBSCAN(eps=0.5, min_samples=5, metric='precomputed')
```

- **`metric='precomputed'`**: a precomputed distance matrix as input

The most suitable eps and min_sample are self-determined, based on general principles and through trial-and-error, instead of blindly using the default values.

- The min_sample value specifies the minimum number of samples in a cluster, and a small value means that DBSCAN will build more clusters from noise points. There is one heuristic approach, the log of the total number of points to be clustered, to determine the knee point.
- The eps specifies the distance range where the core points and border points of a cluster are within that range, while the noise points are far away. The k-distance plot is used to help the analysis with a knee point.
- However, there are sometimes no obvious knees or multiple knees and the heuristics might not be correct. Therefore, trial and error is needed to find suitable parameters.

Agglomerative

Agglomerative Clustering is implemented by AgglomerativeClustering:

```
AgglomerativeClustering(n_clusters=2, affinity='precomputed',
                        linkage='average')
```

- **`n_clusters=2`**: 2 clusters outputted from the clustering
- **`metric='precomputed'`**: a precomputed distance matrix as input
- **`linkage='average'`**: average is used to merge the two closest clusters and update the proximity matrix, since it compromises between max, more resilient to noises but tends to break large clusters, and min, can handle non-elliptical shapes but sensitive to noise and outliers. It is less likely to produce extreme biased results.

Experimental Result

Data Preprocessing

Data Integration

The original dataset from Kaggle comprises of two data: one provides 4803 movie samples, and each contains 20 attributes, the other provides the same movie samples but 4 attributes.

The first movie sample is the movie *Avatar*, which has a popularity of 150.437577 and a vote average of 7.2 calculated from 11800 votes. Some other attributes are listed below:

	id	name
0	28	Action
1	12	Adventure
2	14	Fantasy
3	878	Science Fiction

	name	id
0	Ingenious Film Partners	289
1	Twentieth Century Fox Film Corporation	306
2	Dune Entertainment	444
3	Lightstorm Entertainment	574

	iso_3166_1	name
0	US	United States of America
1	GB	United Kingdom

	iso_639_1	name
0	en	English
1	es	Español

Avatar's genres, production companies, production countries and spoken languages (from left to right)

	credit_id	department	gender	id	job	name
0	52fe48009251416c750aca23	Editing	0	1721	Editor	Stephen E. Rivkin
1	539c47ecc3a36810e3001f87	Art	2	496	Production Design	Rick Carter
2	54491c89c3a3680fb4001cf7	Sound	0	900	Sound Designer	Christopher Boyes
3	54491cb70e0a267480001bd0	Sound	0	900	Supervising Sound Editor	Christopher Boyes
4	539c4a4cc3a36810c9002101	Production	1	1262	Casting	Mali Finn
5	5544ee3b925141499f0008fc	Sound	2	1729	Original Music Composer	James Horner
6	52fe48009251416c750ac9c3	Directing	2	2710	Director	James Cameron
7	52fe48009251416c750ac9d9	Writing	2	2710	Writer	James Cameron
8	52fe48009251416c750aca17	Editing	2	2710	Editor	James Cameron
9	52fe48009251416c750aca29	Production	2	2710	Producer	James Cameron
10	52fe48009251416c750aca3f	Writing	2	2710	Screenplay	James Cameron
11	539c4987c3a36810ba0021a4	Art	2	7236	Art Direction	Andrew Menzies
12	549598c3c3a3686ae9004383	Visual Effects	0	6690	Visual Effects Producer	Jill Brooks
13	52fe48009251416c750aca4b	Production	1	6347	Casting	Margery Simkin
14	570b6f419251417da70032fe	Art	2	6878	Supervising Art Director	Kevin Ishioka
15	5495a0fac3a3686ae9004468	Sound	0	6883	Music Editor	Dick Bernstein
16	54959706c3a3686af3003e81	Sound	0	8159	Sound Effects Editor	Shannon Mills
17	54491d58c3a3680fb1001ccb	Sound	0	8160	Foley	Dennie Thorpe
18	54491d6cc3a3680fa5001b2c	Sound	0	8163	Foley	Jana Vance
19	52fe48009251416c750aca57	Costume & Make-Up	1	8527	Costume Design	Deborah Lynn Scott

	cast_id	character	credit_id	gender	id	name	order
0	242	Jake Sully	5602a8a7c3a3685532001c9a	2	65731	Sam Worthington	0
1	3	Neytiri	52fe48009251416c750ac9cb	1	8691	Zoe Saldana	1
2	25	Dr. Grace Augustine	52fe48009251416c750aca39	1	10205	Sigourney Weaver	2
3	4	Col. Quaritch	52fe48009251416c750ac9cf	2	32747	Stephen Lang	3
4	5	Trudy Chacon	52fe48009251416c750ac9d3	1	17647	Michelle Rodriguez	4
5	8	Selfridge	52fe48009251416c750ac9e1	2	1771	Giovanni Ribisi	5
6	7	Norm Spellman	52fe48009251416c750ac9dd	2	59231	Joel David Moore	6
7	9	Moat	52fe48009251416c750ac9e5	1	30485	CCH Pounder	7
8	11	Eytukan	52fe48009251416c750ac9ed	2	15853	Wes Studi	8
9	10	Tsu'Tey	52fe48009251416c750ac9e9	2	10964	Laz Alonso	9
10	12	Dr. Max Patel	52fe48009251416c750ac9f1	2	95697	Dileep Rao	10
11	13	Lyle Wainfleet	52fe48009251416c750ac9f5	2	98215	Matt Gerald	11
12	32	Private Fike	52fe48009251416c750aca5b	2	154153	Sean Anthony Moran	12
13	33	Cryo Vault Med Tech	52fe48009251416c750aca5f	2	397312	Jason Whyte	13
14	34	Venture Star Crew Chief	52fe48009251416c750aca63	2	42317	Scott Lawrence	14
15	35	Lock Up Trooper	52fe48009251416c750aca67	2	986734	Kelly Kilgour	15
16	36	Shuttle Pilot	52fe48009251416c750aca6b	0	1207227	James Patrick Pitt	16
17	37	Shuttle Co-Pilot	52fe48009251416c750aca6f	0	1180936	Sean Patrick Murphy	17
18	38	Shuttle Crew Chief	52fe48009251416c750aca73	2	1019578	Peter Dillon	18
19	39	Tractor Operator / Troupe	52fe48009251416c750aca77	0	91443	Kevin Dorman	19

Avatar's first 20 casts and crews, in total 83 casts and 153 crews

ID columns in two data frames are aligned and two data are merged on the ID.

Data Cleaning

Inconsistent and irrelevant data are removed in the data cleaning process.

The budget and revenues are important to measure the commercial success of the movies. However, as shown in the histogram below, 1031 samples have 0 budget record. This is unlikely the actual case. The movie The Cat in the Hat, which has a 0 budget in the database, actually has the estimated budget of USD 109,000,000.

By manually comparing with the official websites, It is found that budget is too flawful to be selected and is therefore entirely dropped. Similarly, revenue attribute is entirely dropped as well. And in that case, unfortunately, no inference of the commercial success can be made.

Feature Selection

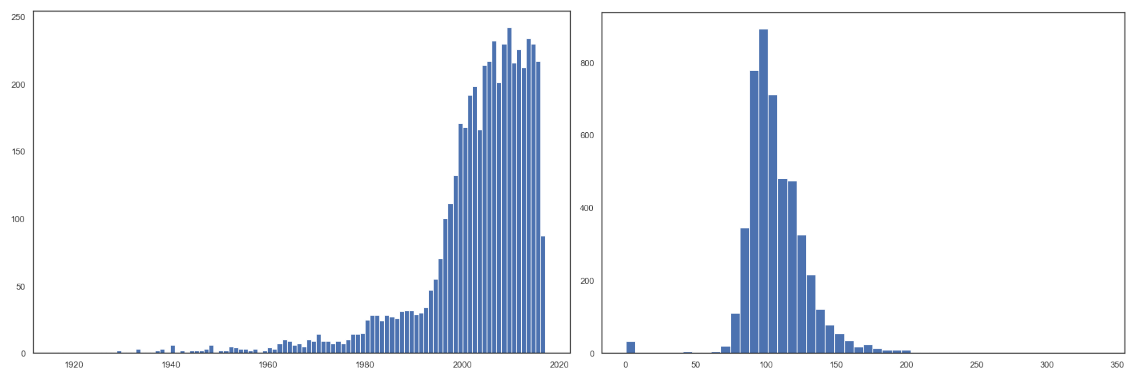
Genres and keywords are carefully inspected here. There are 20 distinct genres found and 5133 distinct keywords. The summary statistics suggests that the top 3 genres and top 5 keywords should be selected.

```
count    4791.000000   count    4791.000000
mean        7.545815   mean        2.534126
std         6.281591   std         1.120335
min         0.000000   min         0.000000
25%         3.000000   25%         2.000000
50%         6.000000   50%         2.000000
75%        11.000000   75%         3.000000
max        97.000000   max         7.000000
```

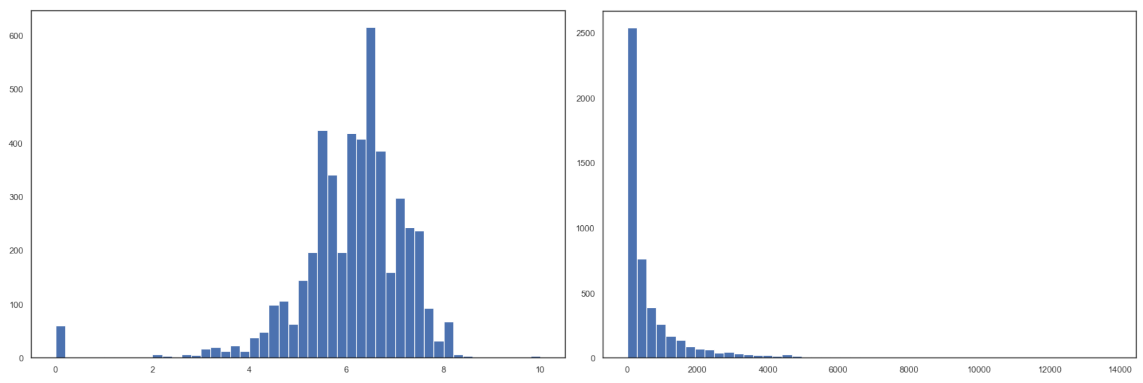
Summary statistics of number of genres (on the left) and keywords (on the right)

Numerical attributes, popularity, release date, runtime, vote count, and vote average have been inspected next. They also reveal some interesting facts:

- September is the month where most numbers of movies are released.
- The runtime is right-skewed.
- The vote average is left-skewed and 60 non-rated movies has vote average of 0.



Histogram of the distribution of release date (on the left) and runtime (on the right)



Histogram of the distribution of vote average (on the left) and vote count (on the right)

Other attributes, original language, original title and spoken languages are duplicated and removed from the data. Feature elimination process has been done to drop the ID columns. The data now has 4790 movie samples, and each contains 29 attributes: id, title, language, overview, popularity, release date, runtime, vote count and average, number of casts, 3 major cast names and genders, number of crews, director name, 3 genres, 5 keywords, 2 major production companies and major production country.

For simplicity for further analysis, 8 most representative features are selected for the data mining, there are 6 columns of categorical data and 2 columns of numerical data.

To perform data mining, the data is converted to vector. Each numeric column is analyzed and divided into several buckets, each bucket being one category, that is categorical data. The categorical data is assigned a unique number.

	language	cast_0	director	genre_0	company_0	country	cast_num	crew_num	popularity
0	0	202	13	5	31	0	0	1	1.0
1	0	371	1212	13	584	6	0	0	0.0
2	26	1690	1956	5	1051	9	0	0	0.0
3	0	32	135	6	344	0	0	3	1.0
4	0	518	1793	8	942	0	0	1	0.0

Data after bucketization and vectorization

Data Exploration

The titles of movies are the most compact summary of the movie. The title length has a right-skewed distribution, with a mean of 15.33. The most frequently used meaningful words are “Man” with frequency of 1.46% and “Love” with 1.17%.

The correlation among the numerical attributes, is an important point based on which the consequent data exploration progresses. Below is the colored correlation matrix.

	popularity	runtime	vote_average	vote_count	cast_num	crew_num
popularity	1	0.224723	0.275023	0.777924	0.3586	0.446948
runtime	0.224723	1	0.376691	0.271726	0.310134	0.218604
vote_average	0.275023	0.376691	1	0.314757	0.230024	0.170541
vote_count	0.777924	0.271726	0.314757	1	0.44263	0.474874
cast_num	0.3586	0.310134	0.230024	0.44263	1	0.351287
crew_num	0.446948	0.218604	0.170541	0.474874	0.351287	1

Note that very high or small vote averages are generally based on small numbers of votes per movie. This suggests that using plain movie vote average as ranking is not reliable. Weighted rating, given by the formula below, are proved to be better evaluation, as it takes the number of votes into account:

$$W = \frac{R \times v + C \times m}{v + m}$$

where W is weighted rating, R is the vote average, v is the vote count, m is the minimum votes required, the 75% quantile 740, and C is the mean vote, 6.09 for this dataset.

With this formula, movies with very few votes, will get weighted ratings close to the mean, and movies with vote counts higher, will get weighted ratings close to their vote average. Some observations in the change of top 20 movies given by the weighted rating:

- *Inception*, the movie with the largest number of votes, was just outside the Top 20 of vote average and has moved up to 7th place.
- *Seven Samurai*, vote count of only 878 but vote average of 8.2, drops out.

Simple movie recommendation applications can be developed from data analytics results.

First recommendation is based on the preference of genre and language. The basic idea behind this recommendation is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.

	id	title	popularity	release_date	country
2184	9470	Kung Fu Hustle	13.658985	2004-02-10	CN
4076	12207	The Legend of Drunken Master	15.046612	1994-02-03	HK
3097	11770	Shaolin Soccer	17.547680	2001-07-12	CN
1357	365222	Ip Man 3	19.167377	2015-12-19	HK
3307	33542	Rumble in the Bronx	17.318839	1995-01-30	HK

5 movies recommended for preference of Chinese action movie

Second recommendation is based on the similarity between movies, in terms of number of common main casts, director, genre, and weighted rating. The basic idea behind is that audiences are more likely to be attracted by movies similar to their favorites.

	id	title	popularity	release_date	country
0	19995	Avatar	150.437577	2009-12-10	US
1652	14164	Dragonball Evolution	21.677732	2009-04-01	US
216	87827	Life of Pi	51.328145	2012-11-20	US
587	2756	The Abyss	24.961625	1989-08-09	US
243	1593	Night at the Museum	48.780039	2006-10-20	US

5 movies recommended for movie favorite Avatar

Classification

K-Nearest Neighbors, Gaussian Naïve Bayes, Support Vector Machines, Artificial Neural Networks, and Ensemble Classifiers predict the popularity from features. Their predictions are evaluated using some standard metrics for performance.

Metrics for Performance

Accuracy Score

Accuracy score simply computes the accuracy of classification results, that is compare the origin popularity with the predicted one, implemented by a method from sklearn.metrics.

Confusion Matrix

Confusion matrix evaluates True Positive, True Negative, False Positive and False Negative of the labels predicted by the classifier, implemented by a method from sklearn.metrics.

Precision

Precision calculates metrics for labels, and find their averages weighted by the support, the number of true instances for each label. This alters macro to account for label imbalance and result in an F-score that is not between precision and recall.

ROC Curve

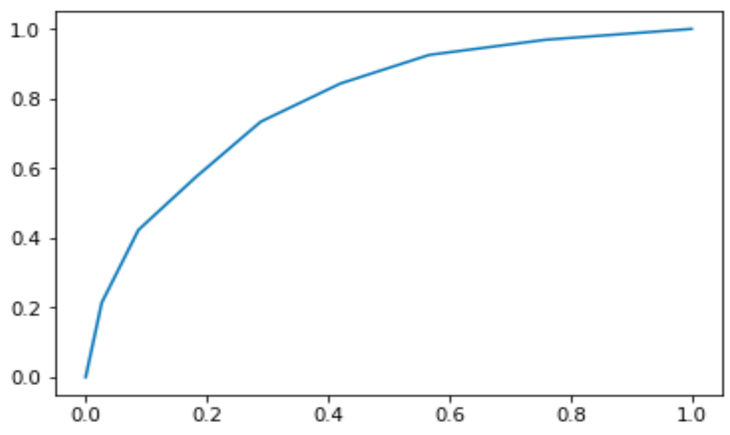
ROC Curve visualizes how well a classifier performs. It requires two arrays, correct labels for test data and probabilities that each record is predicted to be the positive class, and returns the false positive rate, the true positive rate, and the threshold value of a curve.

Result Analysis

Each classifier is evaluated using metrics for performance evaluation.

K-Nearest Neighbors

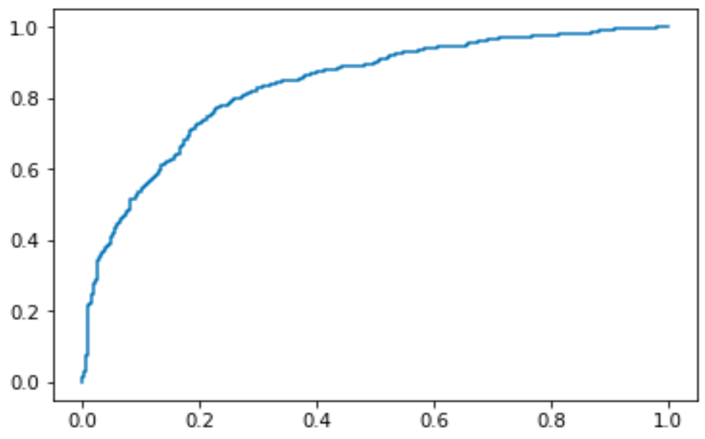
- Accuracy score: 0.72243108
- Confusion matrix: TN 563, FP 229, FN 214, TP 590
- Precision: 0.72247047
- ROC curve:



K-Nearest Neighbors algorithm works via taking majority vote from the class labels of the nearest neighbors. The number of nearest neighbors greatly affects the results' accuracy: a small k would be sensitive to noise points, while a large k would be vulnerable to outliers.

Gaussian Naïve Bayes

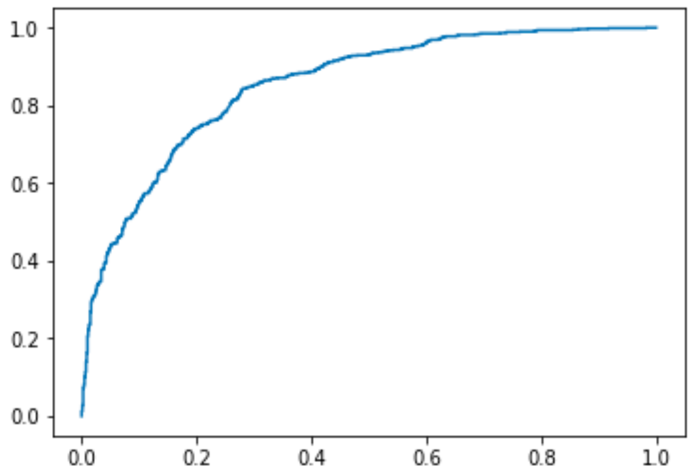
- Accuracy score: 0.74310777
- Confusion matrix: TN 500, FP 292, FN 118, TP 686
- Precision: 0.75484187
- ROC curve:



Gaussian Naïve Bayes considers each attribute consisting of the movie record and class label popularity as random variables. Instead of estimate probability of label, this method applies Bayes theorem, equivalently estimate probability of seeing that set of attributes given the label. Thus, an unseen data is classified to the label maximizing that probability.

Support Vector Machine

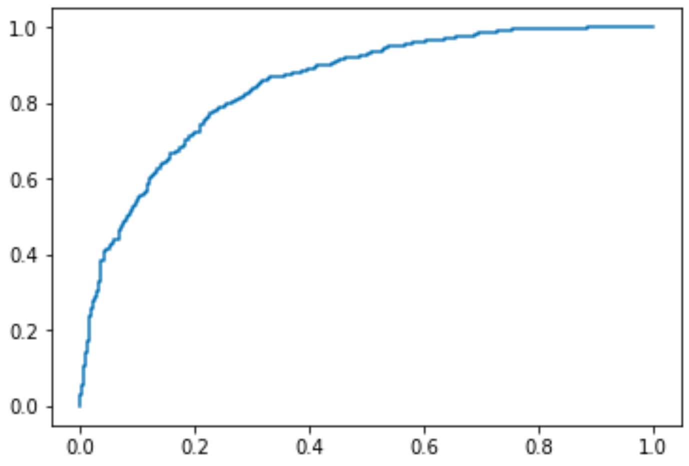
- Accuracy score: 0.77130326
- Confusion matrix: TN 587, FP 205, FN 160, TP 644
- Precision: 0.77207216
- ROC curve:



Support Vector Machine finds the decision boundary that will separate the data.

Artificial Neural Networks

- Accuracy score: 0.77506266
- Confusion matrix: TN 615, FP 177, FN 182, TP 622
- Precision: 0.77508405
- ROC curve:

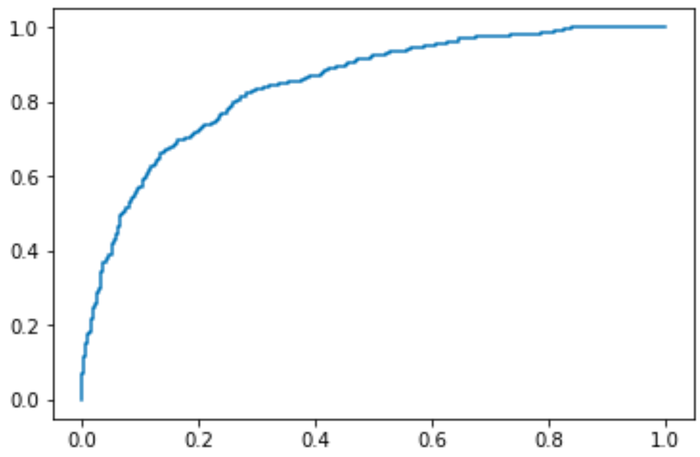


Artificial Neural Networks requires many parameters, including hidden layer sizes, solver, max iterations, and convergence condition. These are found through trial-and-error, taking into account both the performance and the training time. Others simply follow the default.

ANN classifier consists input layer, output layer and hidden neurons, which are activation functions for weighted sum of data, and training ANN is optimizing the weights of neurons, through gradient descent learning.

Ensemble Classifier

- Accuracy score: Hard 0.74874687, Soft 0.76754386, Weightage 0.76002506
  Soft Voting, with highest accuracy, is chosen to implement further evaluation.
- Confusion matrix: TN 562, FP 230, FN 141, TP 663
- Precision: 0.77072186
- ROC curve:



Ensemble Classifier predicts class label of unknown records by aggregating predictions made by multiple classifiers.

Clustering Analysis

Clustering analysis aims to answer that is there any hidden patterns valuable and is it possible to cluster them into several clusters based on their features. If true, some implicit groupings of movies can discover.

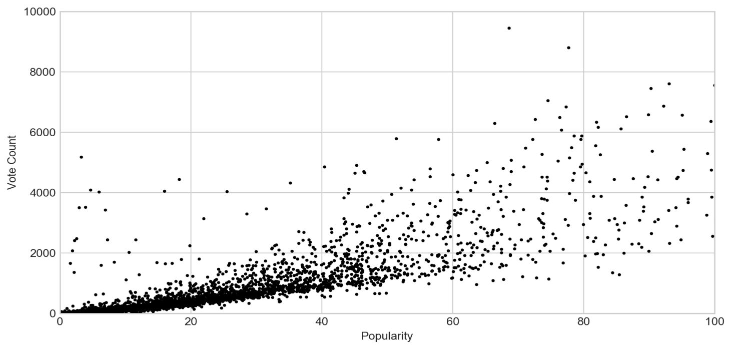
The number of clusters and the number of points in each cluster should first be determined. After obtaining the clusters, various methods, such as cohesion and separation, are used to evaluate the results.

Result Analysis

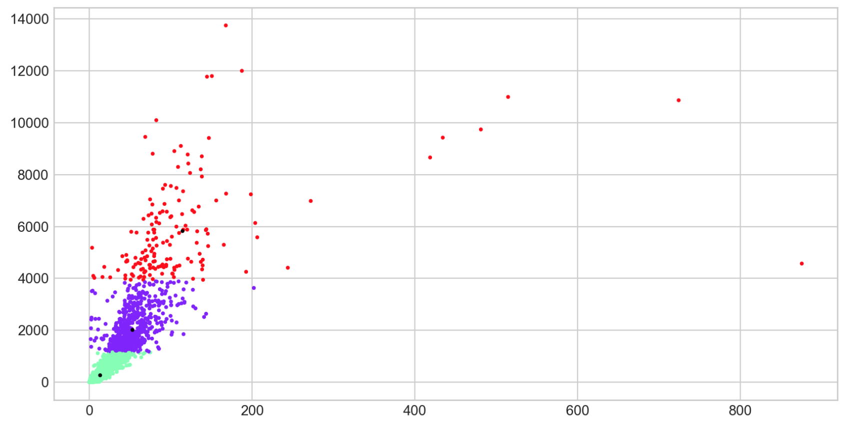
K-Means

Take popularity of movies as x-axis and count of votes as y-axis. The value range of popularity is from 0 to 130 and the value range of vote count is from 0 to 11000. Eliminate outliers and constrain the values to [0, 100] for popularity and [0, 10000] for vote count.

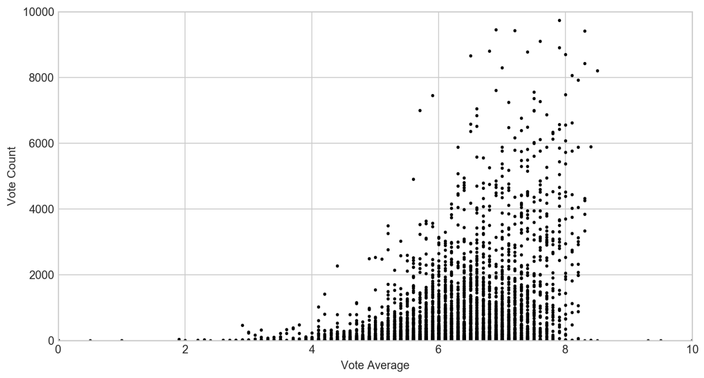
The graph shows that most of the points focus on the area from (0, 0) to (40, 2000). However, the concentration of points results in the occlusion of different values. The details in it cannot be shown clearly without further processing.



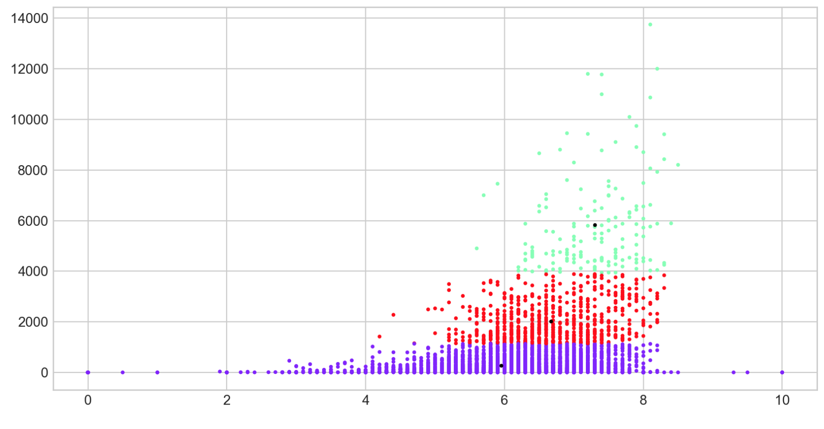
The result graph after k-means clustering with k equal 3 is shown below. The green cluster has smaller area than the purple and red clusters, which means points are gathered at (0,0) to (50,1000) in a more concentrated way. It also shows that movies that are less popular are more likely to have less people to vote for them. In other words, people are more willing to vote for more popular movies.



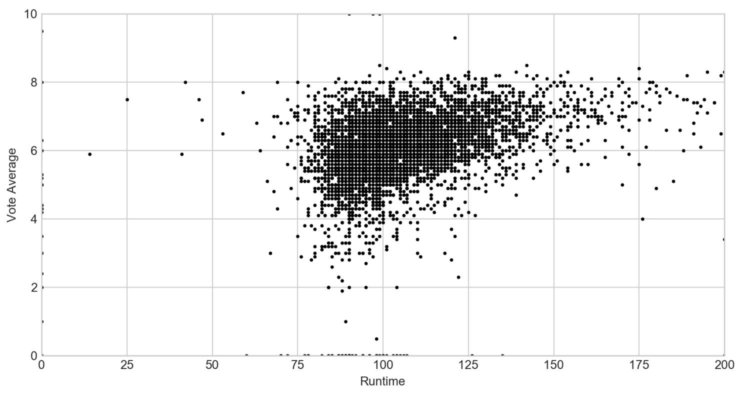
Take the average vote rates as x-axis and count of votes as y-axis. The value range of average vote rates is from 0 to 10. The graph shows that most of the points focus on the area from (6, 0) to (8, 2000), but the concentration of points results in the occlusion of different values.



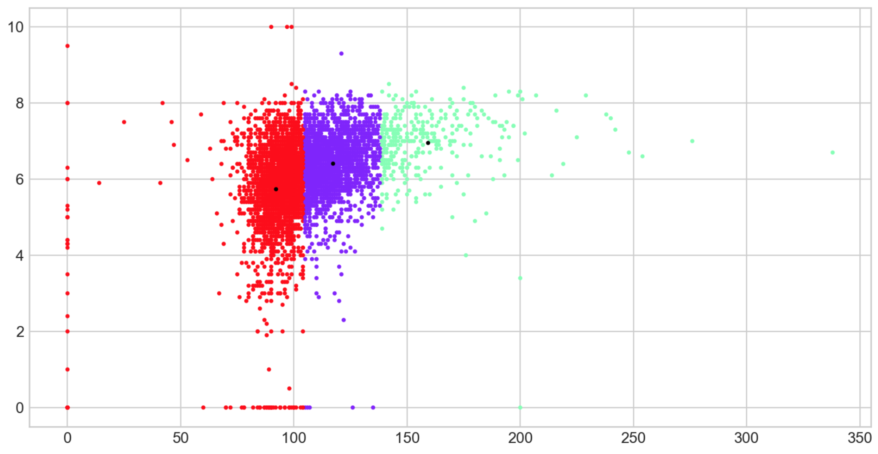
The result graph after k-means clustering of k equal to 3 is shown below. The purple cluster has smaller area than the red and green clusters, which means points are gathered at the area of (6, 0), (8, 0), (6, 1000), (8, 1000) in a more concentrated way. It also shows that most movies usually get the average rate of 6 to 8 and movies with high average rate usually have more votes. In other words, people do not like to vote for those low-rate movies.



Take the runtime of movies as x-axis and average vote rates as y-axis. The value range of runtime is from 0 to 200. The graph shows that most of the points focus on the area from (75, 4) (125, 4) to (75, 8) (125, 8).

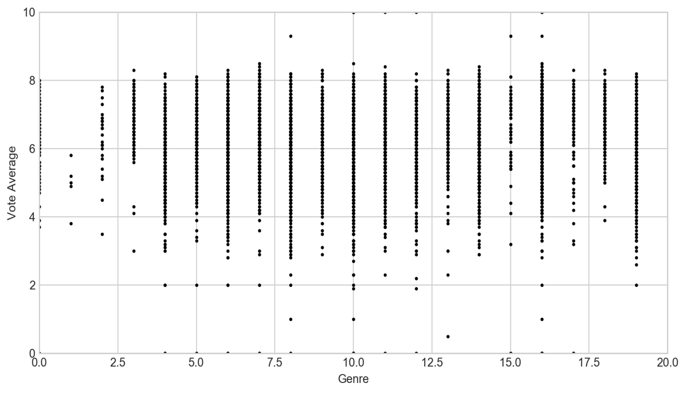


The result graph after k-means clustering of k equal to 3 is shown below. There are many outliers on the y-axis and x-axis which should not be considered as final results. When larger number of clusters is chosen, post-processing is required to eliminate small clusters that may represent outliers. The graph illustrates that no matter what the runtime of movies is, it does not affect their average rates. In other words, there is no evidence showing that longer movies mean better movies.

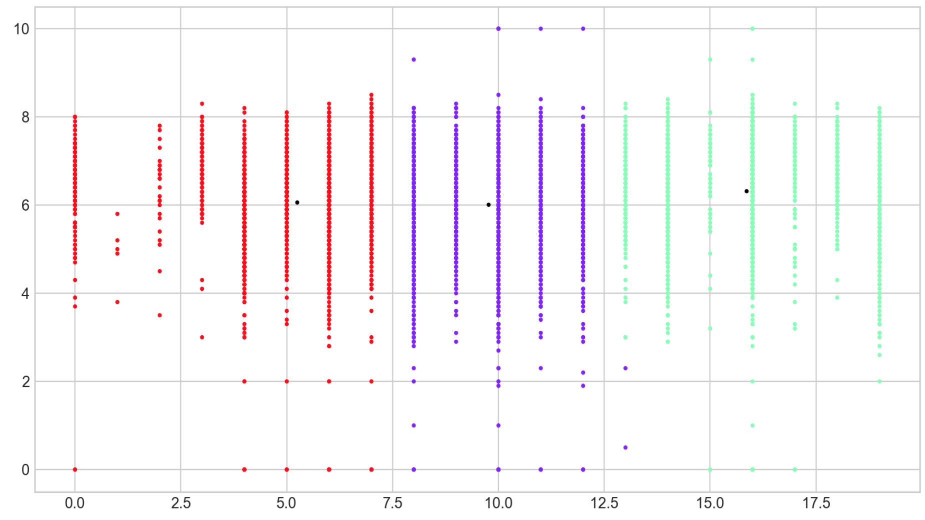


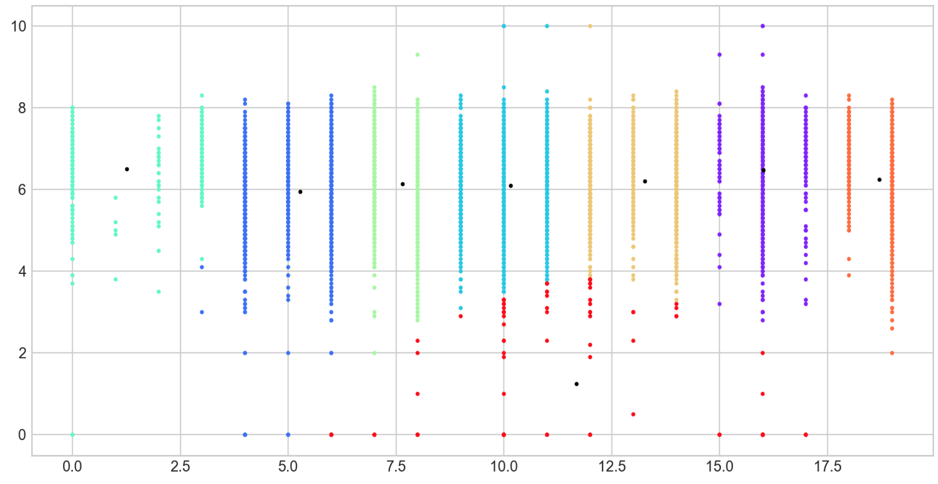
Take genre of movies as x-axis and their average vote rates as y-axis. The genre has already been converted to numerical values, which is from 0 to 19 and 20 genres in total.

There is no concentration of points in this graph since x-axis only represents the index of different types of movies, not numerical values with real meanings. However, because the points only appear on the integer value of x-axis, which is a vertical line, the effect of concentration of points is still significant and results in the occlusion of different points. The details in it cannot be shown clearly without further processing.



The result graph after k-means clustering of k equal to 3 and k equal to 8 is shown below. When k is 8, it shows the distribution of points in a clearer way. The red cluster in the graph with k is 8 should be considered as empty cluster. Post-processing is required to eliminate this sparse cluster that represents outliers. It shows that when people rate for a movie, there is no preference in the genre of movies. The well-known popular movie types such as action, science fiction, drama and comedy, do not have higher average rate than the movies of minority interest. The movies of minority interest only have less rate, but not lower rate.





DBSCAN

- `eps=0.374`:
    - number of clusters: 13
    - number of noises: 4015
- `eps=0.375`:
    - number of clusters: 1
    - number of noises: 639
- `eps=0.38`:
    - number of clusters: 1
    - number of noises: 639

There is a sharp turn at **`eps=0.375`**. When less than 0.375, the DBSCAN algorithm returns the majority of noise points. After 0.375, it returns one single huge cluster.

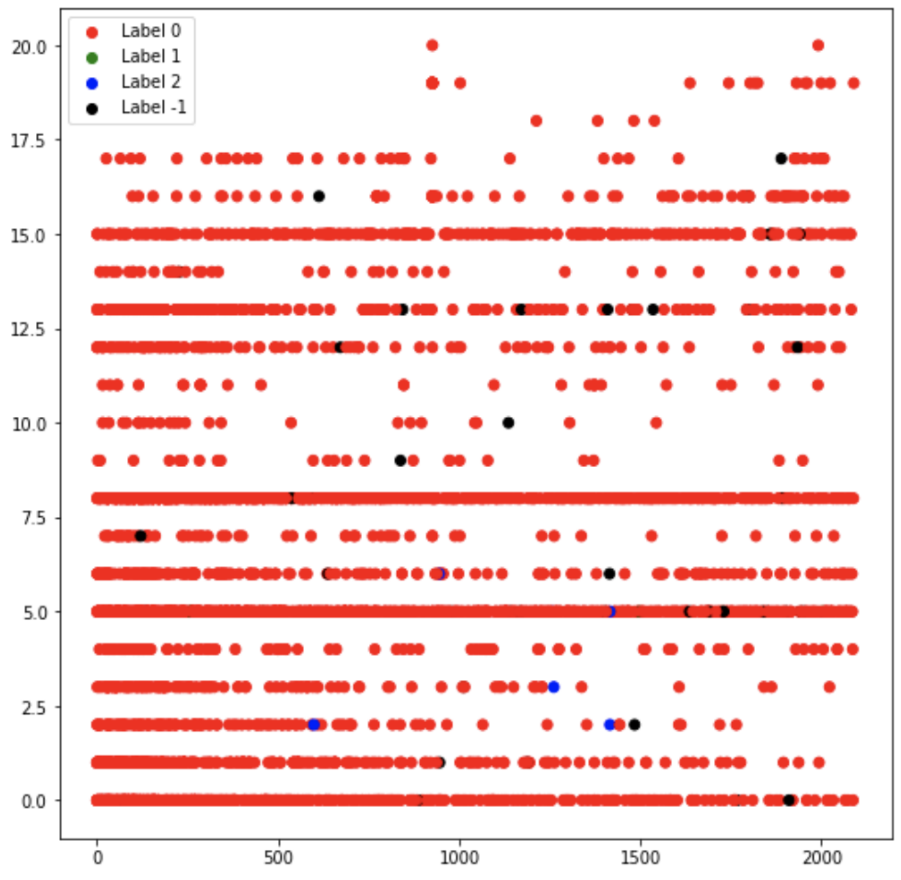
- `min_samples=8`:
    - number of clusters: 1
    - number of noises: 615
- `min_samples=9`:
    - number of clusters: 1
    - number of noises: 639
- `min_samples=10`:
    - number of clusters: 1
    - number of noises: 762

Different values of `min_samples` are tried giving slightly different results.

- `default`:
    - number of clusters: 3
    - cluster 0: 4732
    - cluster 1: 5
    - cluster 2: 7
    - number of noises: 762

DBSCAN with default parameters returns three clusters and very few numbers of noises. This looks a bit more promising. However, the number of points in each cluster is too unbalanced. There is one huge cluster and 2 very small clusters. This is almost the same result compared to DBSCAN clustering results using the previous sets of self-defined parameters.

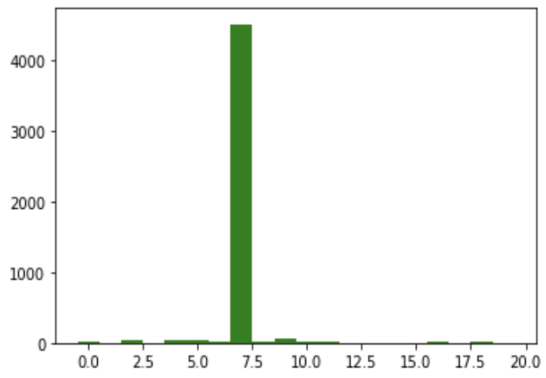
This result agrees with the previous analysis that the clustering fails as the result returned contains a single huge cluster, as shown in the plot below:

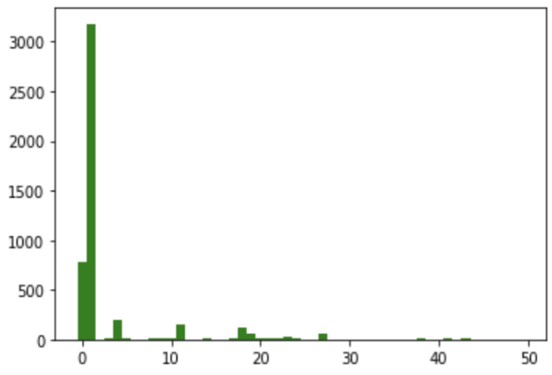


Agglomerative

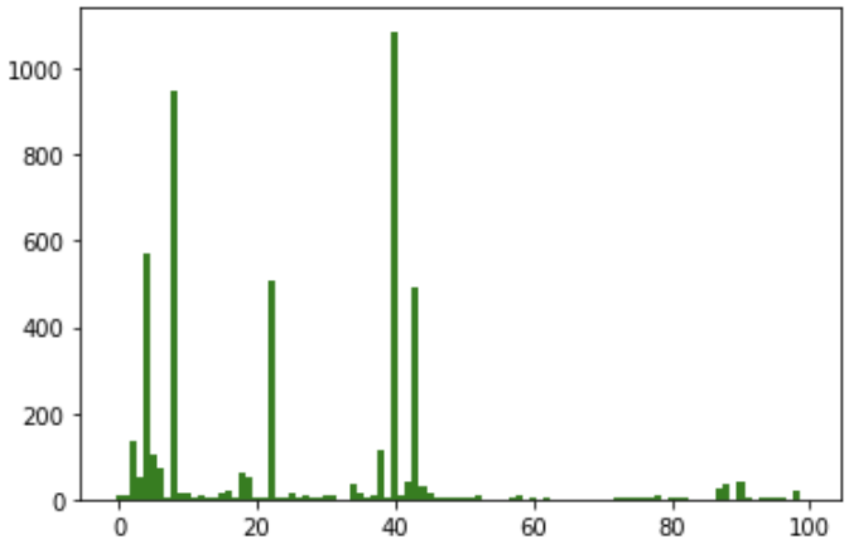
- `default`:
    - number of clusters: 2
    - cluster 0: 5
    - cluster 1: 4785

As shown here, the algorithm returns one very big cluster and one very small cluster with the size of only 5 points. As the number of clusters increases, the results returned by the agglomerative clustering is shown below.





the sizes of 20 clusters (on the left) and 50 cluster (on the right) from Agglomerative clustering



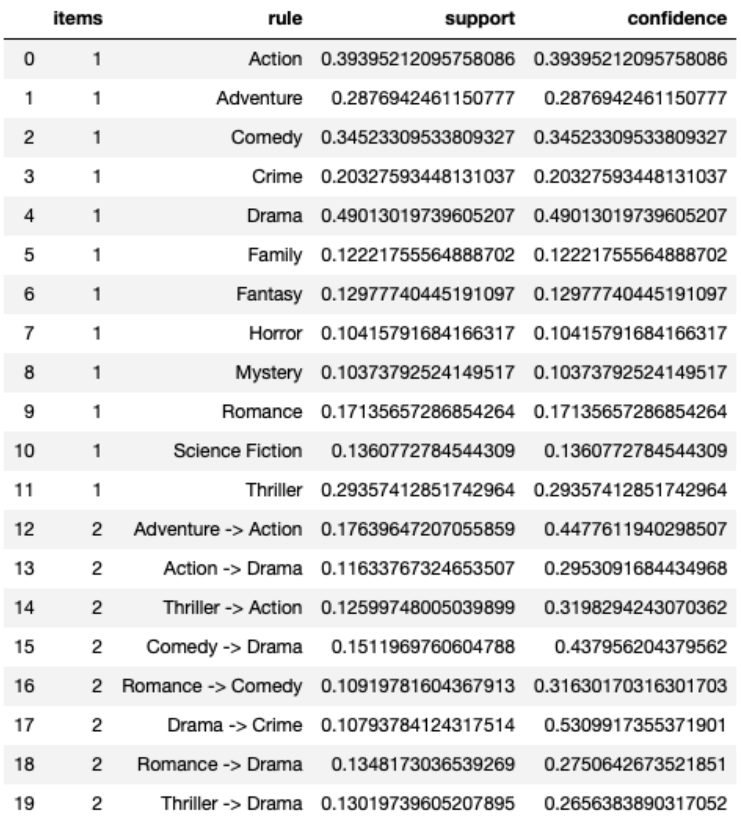
the sizes of 100 clusters from Agglomerative clustering

As shown here, several reasonable-size clusters are returned when the number of clusters gets really huge. Otherwise, it returns one single huge cluster. Therefore, the agglomerative clustering does not return a good result, most likely due to the nature of datasets.

Association Rule

20 association rules are generated for the genres and no association rules are found for the keywords. The rules for genres include 12 rules of single items and 8 of two items. These reveal some interesting patterns, for example adventure and thriller movies tend to involve a lot of action senses, while romance focus more on comedy and drama.

The reasons why no association rules are found for the keywords is that in the data set of 4780 movie samples are 5733 keywords in different languages. Thus, keywords attributes are much more spatial compared to genres which have only 20 distinct types, and association in such spatial data are generally weak.



	items	rule	support	confidence
0	1	Action	0.39395212095758086	0.39395212095758086
1	1	Adventure	0.2876942461150777	0.2876942461150777
2	1	Comedy	0.34523309533809327	0.34523309533809327
3	1	Crime	0.20327593448131037	0.20327593448131037
4	1	Drama	0.49013019739605207	0.49013019739605207
5	1	Family	0.12221755564888702	0.12221755564888702
6	1	Fantasy	0.12977740445191097	0.12977740445191097
7	1	Horror	0.10415791684166317	0.10415791684166317
8	1	Mystery	0.10373792524149517	0.10373792524149517
9	1	Romance	0.17135657286854264	0.17135657286854264
10	1	Science Fiction	0.1360772784544309	0.1360772784544309
11	1	Thriller	0.29357412851742964	0.29357412851742964
12	2	Adventure -> Action	0.17639647207055859	0.4477611940298507
13	2	Action -> Drama	0.11633767324653507	0.2953091684434968
14	2	Thriller -> Action	0.12599748005039899	0.3198294243070362
15	2	Comedy -> Drama	0.1511969760604788	0.437956204379562
16	2	Romance -> Comedy	0.10919781604367913	0.31630170316301703
17	2	Drama -> Crime	0.10793784124317514	0.5309917355371901
18	2	Romance -> Drama	0.1348173036539269	0.2750642673521851
19	2	Thriller -> Drama	0.13019739605207895	0.2656383890317052

20 association rules are generated for the genres

Discussion

Classification

K-Nearest Neighbors

This method is relatively easy to understand as well as to implement. But due to it does not build model explicitly, this can be more expensive than others when doing classification for unknown records. Also, nearest-neighbors classifier makes prediction based on local information, which will be more sensitive to noise.

Naïve Bayes

Naïve Bayes is robust to isolated noise point as well as irrelevant attributes, since the probability can be too small to be ignored for certain instances. Due to this, the accuracy is slightly better than K-Nearest Neighbors. However, this algorithm relies on the independent assumption, which may not always hold. For this set of movie records, this assumption basically holds.

Support Vector Machine

The accuracy of SVC is better than those before. This basically benefits from the nature of this algorithm, which is the solution tends to be the global minimum not a local one compared to k-Nearest Neighbors. And SVC is accurate especially in high dimensional spaces like this case.

Artificial Neural Network

For the dataset used to implementation, ANN classifier has the best performance among all classifiers according to the accuracy it obtains in the performance evaluation. ANN algorithm has a good fault tolerance, corruption of one or more cells of ANN does not prevent from generating output. Also, the unknown record is classified in a global basis thus will largely improve accuracy. However, the training process of ANN is quite long compared with others, also can say the duration of the network is unknown, which may lead to a high time cost.

Ensemble Classifier

Individual classifiers implemented before could be lousy, but the aggregate may give a better classified result. As a result, the result generated by ensemble classifier is better than or at least the same as almost all classifiers.

Clustering Analysis

K-Means

There are several limitations of K-means clustering.

- First, how to select initial centroids is an important aspect that may lead to completely different results. It decides the trends and shapes of every cluster and the Sum of Squared Error will be influenced. Sometimes the initial points will readjust themselves by multiple runs, but sometimes they do not.
- Another factor is the number of clusters. When setting the number of clusters to a larger number than the required k and then selecting from them, it can reduce the effect of terrible initial centroids to some extent.
- At last, postprocessing and preprocessing are also essential to data mining, which can normalize the data and eliminate outliers.

DBSCAN and Agglomerative

To discuss DBSCAN and Agglomerative, the clustering approach fails to explore correlations between these selected attributes and the popularity.
The reason behind such failure might be:

- The relation between these attributes and popularity is too weak and ambiguous so it cannot be found in clusters.
- The data pre-processing is not enough. To solve this problem, more columns should be added into the analysis. For example, more actors list, categories, names of directors etc. What is more, trial and error on different combinations might help.
- The algorithm chosen might affect the result. DBSCAN might not be a good clustering method on this dataset because it is very likely to have varying densities. After adding more attributes into the clustering analysis and achieve some more reasonable results, to improve the accuracy and explore relations, different algorithms could be implemented, such as SNN and Jarvis-Patrick.

Conclusion

Summary of Achievements

A better understanding of this TMDb dataset is achieved through data exploration and clustering analysis of data mining. Using five classification algorithms, the predictions for the popularity of movies can be done with desirable accuracy. Besides, movie recommendations application based on preference and favorite is designed and implemented.

Among five classification algorithms selected, KNN is the fastest but with low accuracy. SVM, ANN, and Ensemble classifier require long training time but obtain much better accuracy. Naive Bayes is somewhere in between. In general, as evaluated with the metrics, it is concluded that ANN has the best performance.

K-Means is useful for globular shapes, converges fast and has low complexity. The relations between features, popularity, vote counts, vote rate and runtime are explored, and some valuable understandings are obtained.

DBSCAN and Agglomerative clustering are later implemented to overcome the fact that K-Means is dependent on initial centroids and sensitive to noise. Both are used to try to cluster the dataset to find some relation between eight selected attributes.

Directions for Improvements

For movie prediction, selection of classification training features can be further optimized, aiming at some further improvement in accuracy.

For K-means clustering, some optimized algorithms like k-medians and k-medoid could be applied for more accurate results and hopefully provides more data insight. And for DBSCAN and Agglomerative clustering, the exploration of possible multi-attribute clustering results can be improved by different combinations of the selected 29 attributes.

Reference

1. Tan, P. (2013). Introduction to Data Mining. Pearson.
2. Vanderplas, J. (2017). Python Data Science Handbook. O'Reilly.