Queensland University of Technology
CAB420 - Machine Learning

Final Project

Movie Rating Prediction

Team No. 30:
Rune Leistad, n9776460, 25%
KA LONG Lee (Eric) , n9845097, 40%
Sean Drury, n9669779, 10%
Jenny Bogen Griffiths, n10405127, 25%

Motivation

The motivation behind this project is the demand for online movie and series providers to tailor their content recommendations to their individual users. Over the last few years there has also been an increase in online providers that produce their own content. Being able to predict the interests of their users would be beneficial when deciding which series or movies to produce next.

The goal for this project is to predict how a user will rate a given movie based on the previous movies the user has rated, as well ratings for similar movies made by other similar users. In addition to this we wish to explore different ways of clustering within the data set. We will present three different approaches that attempts to solve the task of prediction, as well as two clustering approaches.

Related works

The problem of predicting movie ratings is a popular one, and a number of approaches already exist. These approaches are either based solely on previous ratings made by users, or they also take into account each user's features (e.g. age, profession, gender etc.), which is known as content filtering. For the first strategy, neighborhood approaches or latent factor models are the most widely used methods. Another widely used method is collaborative filtering which is a method that "relies on the past preference [of a user]/rating correlation with other users"[1].

Some of the most popular approaches to our problem and the Netflix challenge (which is a similar problem to ours) have been collaborative filtering as already mentioned, k-means clustering[2a] and regression models (softmax[2b], linear, logistic and kernel[3]). Our first idea was to use a clustering method, so we decided on implemented k-means clustering, as well as a hierarchical clustering, to see if we could find similar users or movies within the data. As a solution to the movie rating prediction problem, we have implemented and trained a neural network, a naïve Bayes classifier and a random forest classifier.

Data sets

The small MovieLens[4] data set consists of about 100.000 ratings made by 600 users on 9700 movies between March 29, 1996 and September 24, 2018. The data set consists of four .csv-files which contain genome scores, genome tags, ratings, links and genre tags related to the movies.

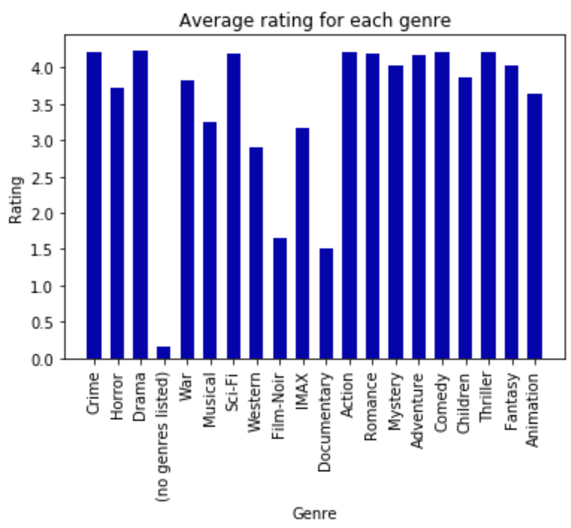
When preprocessing the data we chose to focus only on movies.csv, tags.csv and ratings.csv. The information in these data files was combined to make a dataframes with user IDs, movie IDs, ratings and all the different genres and tags as labels. The dataframe was then modified to only contain the information needed for each method. We also made a dataframe with the average rating each user has given to each genre.

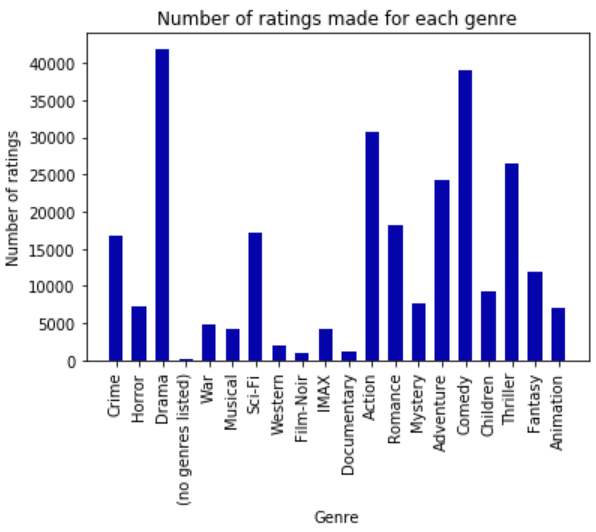
From the figures we can see how the data is biased. Users will generally give high ratings, certain genres have a much higher number of ratings than others, and some genres are generally lower rated than others. This means that two users who for example gave high

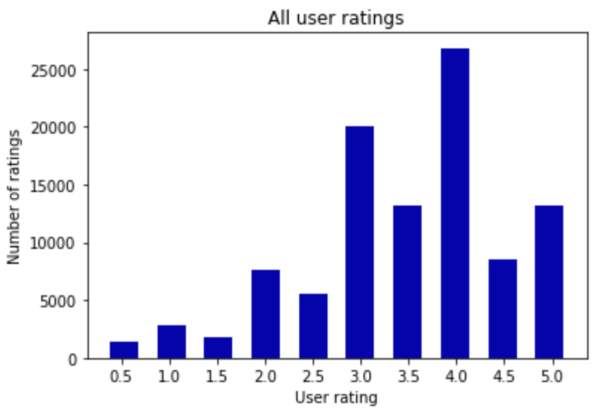
[1] "A Movie Rating Prediction Algorithm with Collaborative ... - IEEE Xplore." http://ieeexplore.ieee.org/abstract/document/5562751/. Downloaded June 7th, 2019.
[2] (n.d.). Movie Recommendations from User Ratings - CS229 - Stanford .... Downloaded June 3rd, 2019 from http://cs229.stanford.edu/proj2013/Bystrom-MovieRecommendationsFromUserRatings.pdf
[3] (n.d.). User rating prediction for movies - Semantic Scholar. Downloaded June 3, 2019 from https://pdfs.semanticscholar.org/e7ae/2379489570302b28f396084b368be64db4ca.pdf
[4] (n.d.). XXXX The MovieLens Datasets: History and Context - GroupLens. Downloaded June 3rd, 2019 from http://files.grouplens.org/papers/harper-tiis2015.pdf

ratings to documentaries will probably be more similar than two users who gave high ratings to a drama movie, as drama is a popular genre with a high average rating across all users.

The data set was split into 80/20 or 70/30 training and test data depending on the method used. When using cross validation to tune the chosen hyperparameters for the different models, several values for the number of folds were used to find the best one.





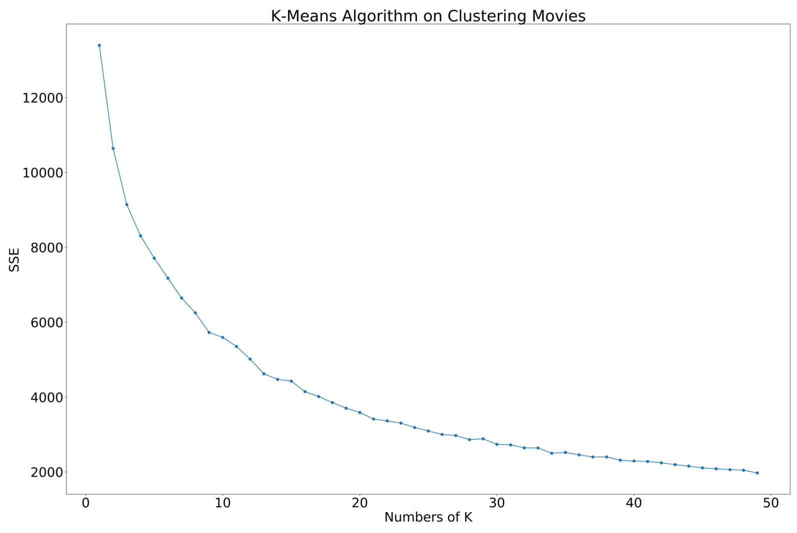


Clustering the data set

We began the project with implementing two different clustering models to get to know the data set and try to discover underlying structures of similar users and movies. The methods that we used were k-means clustering and hierarchical clustering with different input data sets for each method. K-means clustering was used to cluster similar movies based on the genres they are categorised as, while the average rating each user has given to movies within each genre was used as the data set for the hierarchical clustering model.

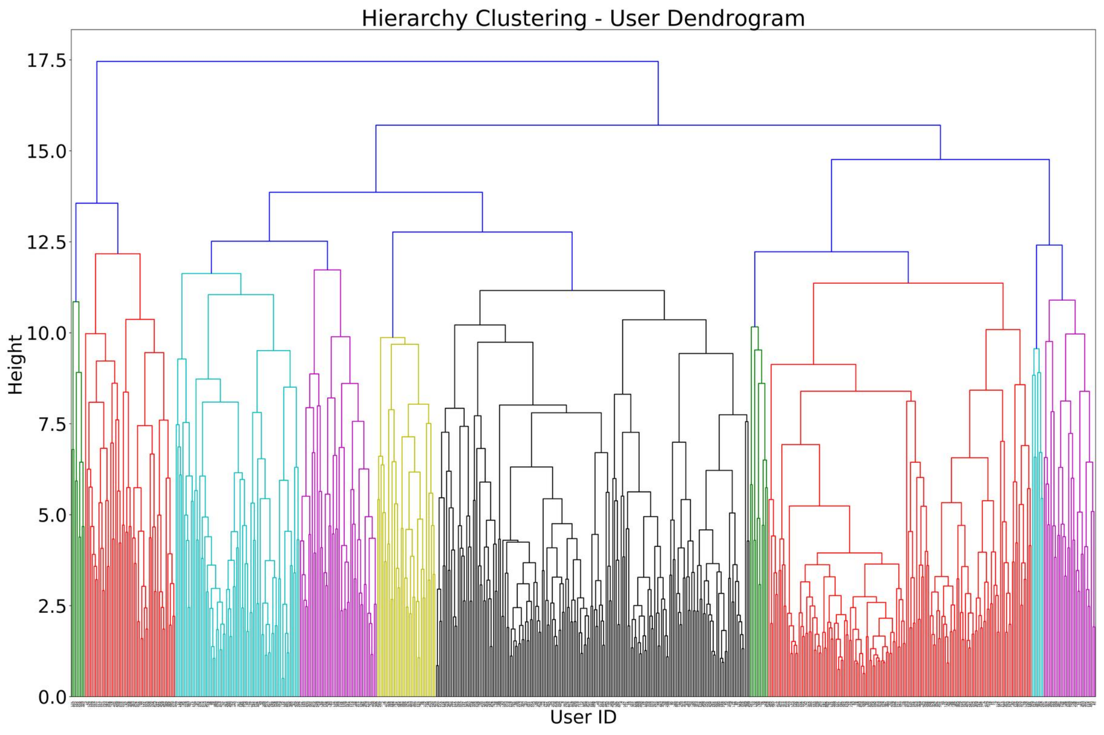
K-means clustering

A K-means clustering method was used to group together similar clusters of movies, this was done based on genres the movies were categorised in the MovieLens data set. The following is a diagram of the K-means algorithm on clustering movies with number of Ks on the x axis and the sum of squared error on the y axis.



Hierarchical clustering

A hierarchical clustering method with complete linkage was used to assign similar users into different clusters based on their average ratings within each genre. As there are twenty genres in the MovieLens data set, the input to the clustering algorithm had twenty features for each user. The linkage resulted in the following dendrogram:



Prediction models

Random forest classifier

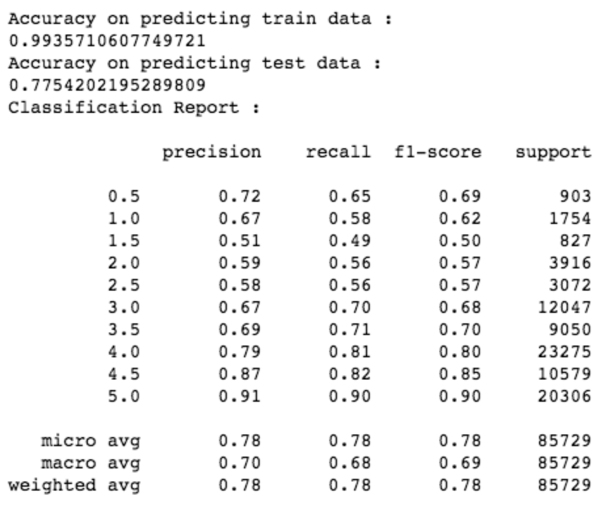
Method

A random forest classifier is an ensemble learning method. The idea behind these methods is to combine several learning algorithms of either the same or of a different kind to improve the prediction performance compared to the performance of just one single model. A random forest classifier uses only one machine learning model, namely a decision tree model. It creates a set of decision trees from random subsets of the data and then decides the final output based on a vote between all the trees. In this case the algorithm uses averaging to decide the final output.

For our random tree model we use tags, genres, and average rating of each movie and genre as our data set, which is then shuffled and split into training and test set with a 70/30 split.

Results

During the course of working on this project, we also tried to implement just a single decision tree classifier. Using this method to predict ratings for the test data resulted in a score of around 0.29 and a RMSE of 1.02 on a rating scale from 0.5 to 5 after trying multiple cross validation methods and multiple tree depths. In other words, it did not perform well at all. The random forest classifier on the other hand, showed much better results as shown in the final accuracies and classification report:



```
Accuracy on predicting train data :
0.9935710607749721
Accuracy on predicting test data :
0.7754202195289809
Classification Report :

              precision    recall  f1-score   support

         0.5       0.72      0.65      0.69       903
         1.0       0.67      0.58      0.62      1754
         1.5       0.51      0.49      0.50       827
         2.0       0.59      0.56      0.57      3916
         2.5       0.58      0.56      0.57      3072
         3.0       0.67      0.70      0.68     12047
         3.5       0.69      0.71      0.70      9050
         4.0       0.79      0.81      0.80     23275
         4.5       0.87      0.82      0.85     10579
         5.0       0.91      0.90      0.90     20306

   micro avg       0.78      0.78      0.78     85729
   macro avg       0.70      0.68      0.69     85729
weighted avg       0.78      0.78      0.78     85729
```

The reason the random forest classifier performs so much better than just a single decision tree is that a single tree may be prone to be affected by noise and outliers[5]. The effects of these error sources is reduced when using several trees instead of one, as many of them will correctly predict the data so that we will get more stable results overall.

[5] "predicting movie ratings - DiVA portal." 7 jul.. 2015, http://www.diva-portal.org/smash/get/diva2:821533/FULLTEXT01.pdf. Downloaded June 7th, 2019.

Naïve Bayes

Method

Naive Bayes is also a statistical classification technique in supervised learning algorithms based on Bayes Theorem. This classification method is also used to predict how a user will rate a given movie. Naive Bayes algorithm assumes that the effect of a particular feature in a class is independent of other features. In comparison with Random forest classifier, the Naive Bayes algorithm can predict the result in a high speed on large datasets.

For our Naive Bayes Gaussian model, we use tags, genres, and userId as our data set, which is then shuffled and split into training and test set with an 80/20 split.

Results

The result shows that the accuracy of Naive Bayes Algorithm is lower than the Random Forest Algorithm. Because of the assumption of independent features, it is almost impossible that model will get a set of predictors which are entirely independent. For instance, most of the user gives a high rating for action movies, but few of the user does not like an action movie. It is obvious that the features in the input vector are not completely independent. Therefore, Random forest Algorithm performs better than Naive Bayes Algorithm in this case.

```
Test data Size :
0.2
Trainning Naive Bayes Classifier ...
Accuracy on predicting train data :
0.2737818721047728
Accuracy on predicting test data :
0.2701695449057792
Classification Report :
              precision    recall  f1-score   support

         0.5       0.00      0.00      0.00       591
         1.0       0.00      0.00      0.00      1196
         1.5       0.03      0.02      0.02       570
         2.0       0.00      0.00      0.00      2758
         2.5       0.00      0.00      0.00      2080
         3.0       0.00      0.00      0.00      8009
         3.5       0.15      0.08      0.11      5971
         4.0       0.28      0.40      0.33     15404
         4.5       0.00      0.00      0.00      6924
         5.0       0.29      0.65      0.40     13650

   micro avg       0.27      0.27      0.27     57153
   macro avg       0.07      0.12      0.09     57153
weighted avg       0.16      0.27      0.19     57153
```

Neural network

Method

A neural network (NN) is a classification technique that is roughly modelled after how a brain is connected and process data. This classification method is used in this report to predict the rating a given user will rate a movie.

The implementation that we went for is a fully connected NN, unique for each user. We ended up separating each user because we felt this would be the most accurate way of estimating a rating for each user. The input to the network was the binary representation of genres given by a movie id. 1's represent that a movie had that specific genre and 0 that it didn't.
Our input layer was fully connected to a hidden layer with the same amounts of neurons as the input layer. This was again connected to our output layer, each output neuron representing a rating between 0.5 and 5.0

We ran a few different tests to see what parameters gave us the highest accuracy rating on both the training and test sets:

- What ratio of training data to test data to use
- Batch size
- Number of epochs

Results

Number of epochs:
We did a test with 10 users, and a test with 5 users. From the average of those two we got an estimated time to run the test to be a little more than an hour. Testing the entire set, with 610 different neural networks, with 20 epochs, took xxxx seconds. A higher number of epochs could have provided more accurate rating predictions, but we were unable to test them due to time constraints.

Testing batch size:
Because of the small number of ratings registered for each user, an average of 165.30~, it was necessary to chose the smallest number of batch size possible to increase the accuracy. The smallest possible was 1.

Accuracy of the neural network:
To give a sense the accuracy of the NNs we created, we chose to show the data size on the x-axis and the accuracy for that data size on the y-axis. On this data we ran a linear regression function to display the relationship between data size and accuracy.

Mean squared error on neural network:
Due to a small number of ratings for each user, this data is very varied in it's output.

Discussion of results

Present the results of all your approaches clearly, and compare them with existing approaches. Discuss why your methods are working better/worse than the existing approaches.

In this project, we used K-means clustering to group similar clusters of movies together. It provides an efficient way to cluster the movies into different groups in a tight way. Although the K Values is hard to predict at the first time, we generate the SSE line chart to display the SSE value changed from K = 0 to K = 50. It helped us to decide the optimal K-value to produce tighter clusters on movie data. In Comparison of Hierarchical clustering, K-Means most of the times computationally faster than it. In addition, there is a total of 193600 movies in the movie.csv. If we use Hierarchical clustering to cluster the movie, the dendrogram will become very complex and squeeze. Therefore, we believe that K-means clustering is the best choice for grouping similar clusters of movies together.

Considering the business perspective, we used Hierarchical clustering to cluster the user's data instead of K-means clustering. Because Hierarchical clustering can better visualize how the users can be a cluster in the dendrogram. There are 600 users only in the users.csv file. Therefore, we can produce clear dendrogram. From the dendrogram, movie company can decide the height to cluster user into how many groups according to their business need. If we want to produce tighter clusters on users data, we also can implement a K-Means SSE chart for assisting us to decide the height value.

As for the neural network, our approach was to create a unique network for each user. Our results shows quite a low accuracy rate, though better than the naïve Bayes for most networks. We believe this is due to the small amount of rating entries in the dataset we chose. This could also be the case for a larger dataset, and so we think creating a separate network for each user was unnecessary. A better approach would be to find some way of integrating the user in to one single neural network. This way you'd have a lot more data entries to learn from and this would in turn give a lot better results.

We were also a little hindered by our computing power. For our relatively short period of testing, running tests for an hour each time was a bit too time consuming. We could have gotten more tests done with fewer epochs, but also less precise.

Future works

In the next phase, we hope that we can gather extra information related to all the movies in a movies.csv file. Such as director, years, actors and film production company information. It might able to improve the random forest algorithm and make the prediction robust. We also need to add cross-validation of hyperparameters to improve accuracy on different classification method.

Using MovieLens 10M Dataset instead of small MovieLens dataset is also a new challenge for our project. The shortcomings of the Hierarchical clustering is that it can not produce a good looking dendrogram when there are too many examples. If the number of users grows from 600 to 72,000, the dendrogram will become very complex and squeeze. Therefore, it might need to use K-Means to cluster the users instead of Hierarchical clustering when the dataset changed to MovieLens 10M Dataset. On the other hand, Random forest classifier takes 11 minutes to train the classifier when the n_estimator set to 100 and takes 70% train data from small MovieLens dataset. The rating grows from 100k to 10M. In mathematical assumption, it will take 18.3333333 hours to train the classifier.