NETFLIX Customer Rating Prediction with KNIME

Serdar MUTLU,Muhammed Emre SEVUK, Serap YALÇIN
Istanbul Sehir University
34865 Dragos, Istanbul, Turkey
{serdarmutlu, muhammedsevuk, serapyalcin}@std.sehir.edu.tr

03.01.2019

Abstract

In today's world, intense competition environment is expanding and companies desire to retain their customers. As a streaming service, Netflix is also in the competitive field. While customer relationship activities get up to date for a competitive advantage, customer analytic comes into prominence. Thus, customer analysis and recommendation systems are quite significant in order to provide better services and prevent customer loss. Thanks to machine learning algorithms, various models could be built up. In this study, several algorithms are used for recommendation systems, predicting customer behaviour and evaluated their performances.

1 Introduction

Level of aggression and competition is constantly increasing in business life. Every day, dynamic of conducting business is reshaping by innovations and firms. Service industry is one of those industries who experienced a rapid change. New technologies and players put old players in the market in a difficult situation. For a very long time, getting new customers and expanding their spread was the number one priority of companies. However, it has been started to shift toward focusing on their existing customers rather than acquiring new ones. Like many other industries, service industry also lean on customer experience and satisfaction.

Netflix is a firm that provides streaming service allowing their members to watch a wide variety of award-winning TV shows, movies, documentaries, and more on thousands of internet-connected devices. Thanks to Netflix, customers can enjoy unlimited ad-free content.[1]

Netflix is studying the behaviour and the rating of customers to develop method to predict the ratings and set recommendation systems. Subsequently, we aimed the same purpose; establishing a model to predict the all ratings the customers gave the movies in the qualifying dataset based on the information in the training dataset.

In the project, PYTHON is used to make data preparation and KNIME is used as an analysis tool. Main algorithms are applied on KNIME beacause it provides modular computational environment that allows easy visual assembly, interactive data analysis, and data processing. It is also an open source predictive analytics platform. [2]

2 Literature Review

Hsu, Ping-Yu and Shen, Yuan-Hong and Xie, Xiang-An studied user rating prediction models by using classification technique. The forecast of neural networks prediction model is observed as the most accurate in this study.[3]

Several studies are conducted to get better recommendation systems. ARSYS-article recommender system that integrates content based, collaborative and metadata recommendations allows users to select the method that best suits their needs. Keywords are used to find similar articles and Collaborative filtering is applied.[4]

Katarya, Rahul and Verma, Om discussed about a novel recommender system that use of k-means clustering by adopting cuckoo search optimization algorithm applied on the Movielens dataset. Then, mean absolute error, standard deviation, root mean square error and t-value are used as evaluation metrics.[5]

3 Proposed Methodology

To analysis the dataset and summarize its features, exploratory data analysis is conducted.

3.1 Data Identification

Netflix released a movie rating dataset in 2006. Then they challenged the some machine learning related communities to develop best system.[6]

The training data file contains 17770 files, one per movie. The first line of each file contains the movie id followed by a colon. It includes CustomerID, Rating, Date

respectively. Th MovieIDs range from 1 to 17770 and CustomerIDs range from 1 to 2649429. The ratings are in range 1 to 5.

There is another movies information file. It includes MovieID, YearOfRelease and Title of the movies.

Additionally, there is a qualifying dataset for the Netflix Prize. It consists of lines indicating a movie id, followed by a colon, and then customer ids and rating dates, one per line for that movie id. The movie and customer ids are contained in the training set but the ratings are withheld.

The general format of the data that are provided from Netflix is not suitable with KNIME. So, some process are conducted in PYTHON to get better format to use KNIME efficiently.

ID	Year	Title	Runtime	Rating	Directors	Writers	Production Co.	Genres
22	2000	Chump Change	89	5.9	Stephen Burrows	Stephen Burrows	The Burrows of Hollywood	Documentary-Animation-Family
150	2001	Firetrap	97	5.3	Harris Done	Diane Fine-Richard Preston	City Heat Productions	Family

Table 1: Sample Data from IMDB

In addition to these, some data from IMBD is gathered. The dataset includes runtime of the movies, IMDB ratings, name of the directors, writers, production companies and genres as shown in Table 1. The whole datasets are combined in PYTHON before data processing process on KNIME.

3.2 Data Reduction

Data reduction consists of some operations like cleaning, splitting, slicing and mapping. These are necessary so that performing effectively with main data.

There are 4 different dataset that are provided by Netflix. All of these dataset are converted to required format. Then, these are combined to be used in the project. After that, other dataset which is taken from IMDB is added.

For this study, there is no duplicated values but some broken values. Moreover, the dataset is incredibly large to apply some algorithms. So, the dataset is sliced. Some of the movies are deleted from the dataset because they are not common on the dataset which means they are not popular and may be ignored. Additionally, customer who have small number of reviews are removed too because they are not active customers. The removing processes helped to get smaller dataset and improve the the statistical significance.

After cleaning data, there should be made some transformation on Genres since it is a categorical feature. There are lots of categories on Genres column. Because of this, the KNIME gave some memory errors when OneHotEncoder is applied. Then, MultiLabelBinalizer is used instead of OneHotEncoder. Details of the data preparation process are shown in APPENDIX A. After multi label binalizer processing, main data are merged with movie data to use in KNIME.

The last step for data reducing is sampling. Sampling is made for both train and test dataset since number of rows is able to complicate processing and modelling. Last quarter of 2005 data are used because of the capacity restrictions. Using whole dataset will give more accurate results.

3.3 Data Processing

There are 2 main data sets, and 1 supporting data set. Main data sets are Train Data, and Test Data. These sets are including movie, user and rating information. These sets are subsets of main historical data provided by Netflix. Supporting data set is a simple set generated dynamically from movie csv, to be able get Multi Label Binary values for a specific movie.

There are two phases in data processing before model runs: Train data processing, and test data processing.

Both phases follow same processing. Similarity sets are created, and merged based on movie-users similarity, user-movie preferences, and user-users similarity.

Movie-users similarity:

Movie is watched by which users, and what are ratings?

User-movie preference:

User has watched which movies, and what are the ratings? In other words, movie preferences of this specific user.

User-users similarity:

Which users have same movie preferences as our specific user?

3.4 Modelling

Several classification algorithms are established on KNIME according to serve the purpose of the project. In the study, three algorithms have been studied which are Linear Regression, Simple Regression Tree, and Random Forest. All these algorithms are used for modelling regression and prediction of customer rating for a specific movie. Training dataset which covers 80% of data, and test data set which covers 20% of data is used for learning and a simple dataset generated dynamically from movie csv is used for prediction in all models.

Simple Regression Tree:

The procedure follows the algorithm described by ”Classification and Regression Trees” (Breiman et al, 1984), whereby the current implementation applies a couple of simplifications, e.g. no pruning, not necessarily binary trees, etc. The currently used missing value handling also differs from the one used by Breiman et al, 1984. In

each split the algorithm tries to find the best direction for missing values by sending them in each direction and selecting the one that yields the best result (i.e. largest gain). The procedure is adapted from the well known XGBoost algorithm.

Figure 1: Performance of Simple Regression Tree Model

As shown in Figure 1, the model gave negative R square value. Then, MAE and other metrics are not good as needed. So, other algorithms are applied to get better results.

Linear Regression Learner:

Performs a multivariate linear regression. Select in the dialog a target column (combo box on top), i.e. the response. The two lists in the center of the dialog allow you to include only certain columns which represent the (independent) variables.

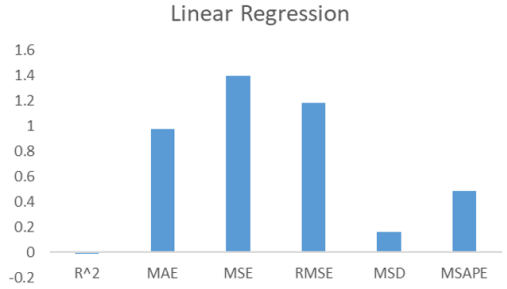


Figure 2: Performance of Linear Regression Model

As shown in Figure 2, the model gave better results than Single Regression Tree. Another algorithm is applied to improve results.

Random Forest Learner Regression:

Learns a random forest, which consists of a chosen number of decision trees. Each of the decision tree models is learned on a different set of rows (records) and a different set of columns (describing attributes), whereby the latter can also be a bit-vector or byte-vector descriptor (e.g. molecular fingerprint). The row sets for each decision tree are created by bootstrapping and have the same size as the original input table. For each node of a decision tree a new set of attributes is determined by taking a random sample of size sqrt(m) where m is the total number of attributes. The output model describes a random forest and is applied in the corresponding predictor node using a simply majority vote.

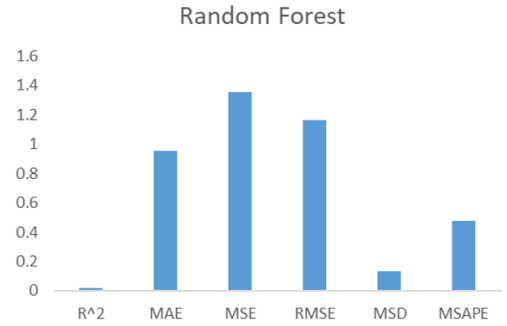


Figure 3: Performance of Random Forest Model

As shown in Figure 3, the best results are gotten among these three applied algorithms. Even the project can be improved with more data and additional features, the performance metrics are sufficient for the aim of the project.

4 Results

In the project, three different algorithms are applied to the dataset. As shown in Figure 4, Random Forest is the best model for the project with lowest MAE and MSE values.

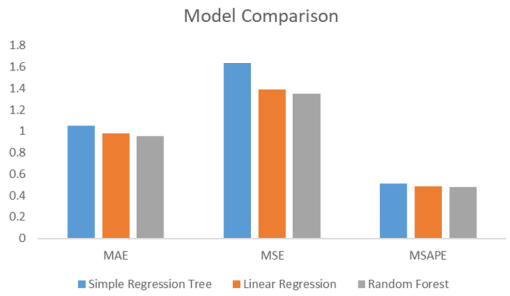


Figure 4: Comparison of Performance

It can be also meaningful to compare predicted rating with specific movie's overall rating, and user's overall rating for all movies user has watched. It should be noted that "Movie Overall Rating"s scale is 10, while "User Overall Rating", and "Predicted Rating" have scales 5. So, as shown in Table 2, Predicted Rating is very close to "Movie Overall Rating", and "User Overall Rating" with same scale.

As mention above, it could be said that random forest is have an advantage via its principle about randomness. In-

	User	User Overall Rating	Movie	Predicted Rating	Movie Overall Rating
S. Regression T.	486644	3.098947368	8	3.146042363	5.4
L. Regression	486644	3.098947368	8	3.147548708	5.4
Random Forest	486644	3.098947368	8	3.14006342	5.4

Table 2: Comparison of Output from Test Part

deed, it is able to generate lots of decision-making mechanism and this mechanisms train with lots of random samples selected from dataset. Thus, random forest model gives one of the best results in terms of performance metrics, which indicates that prediction of the ratings could be made with random forest successfully.

5 Conclusions

As provided from the previous part, with random forest model, it is possible to predict the rations with 95% MSE.

The outputs of this study can be used to provide potential guidance strategies for related divisions. For instance, the marketing division can used the outputs to prepare action plans for recommendation. NETFLIX, which is the provider of the dataset, can use the outputs of the study ant provide significant direction and support to develop and advance in such areas.

References

[1] "MS Windows NT kernel description." https://help.netflix.com/tr/node/412. Accessed: 2019-12-28.

[2] L. Feltrin, "Knime an open source solution for predictive analytics in the geosciences [software and data sets]," *IEEE Geoscience and Remote Sensing Magazine*, vol. 3, pp. 28–38, 12 2015.

[3] P.-Y. Hsu, Y.-H. Shen, and X.-A. Xie, "Predicting movies user ratings with imdb attributes," pp. 444–453, 10 2014.

[4] C. Bancu, M. Dagadita, M. Dascalu, C. Dobre, S. Trausan-Matu, and A. M. Florea, "Arsys-article recommender system," pp. 349–355, 09 2012.

[5] R. Katarya and O. Verma, "An effective collaborative movie recommender system with cuckoo search," *Egyptian Informatics Journal*, vol. 18, 11 2016.

[6] J. Bennett, S. Lanning, and N. Netflix, "The netflix prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

APPENDIX A

Data prepation

There are 4 huge files provided for rating. First merge them as one

```
import pandas as pd
```

```
start = datetime.now()
if not os.path.isfile('data.csv'):
    data = open('data.csv', mode='w')

    row = list()
    files = ['input/combined_data_1.txt', 'input/combined_data_2.txt', 'input/combined_data_3.txt',
'input/combined_data_4.txt']

    for file in files:
        print("Reading ratings from {}...".format(file))
        with open(file) as f:
            for line in f:
                del row[:]
                line = line.strip()
                if line.endswith(':'):
                    movie_id = line.replace(':', '')
                else:
                    row = [x for x in line.split(',')]
                    row.insert(0, movie_id)
                data.write(','.join(row))
                data.write('\n')
        print("Done.\n")
    data.close()
print('Time taken :', datetime.now() - start)
```

```
print("creating the dataframe from data.csv file..")
df = pd.read_csv('data.csv', sep=',', names=['movie', 'user', 'rating', 'date'])
df.date = pd.to_datetime(df.date)
print('Done.\n')

print('Sorting the dataframe by date..')
df.sort_values(by='date', inplace=True)
print('Done..')
```

```
# Find movies, and users with less reviews
f = ['count','mean']

df_movie_summary = df.groupby('movie')['rating'].agg(f)
df_movie_summary.index = df_movie_summary.index.map(int)
movie_benchmark = round(df_movie_summary['count'].quantile(0.7),0)
drop_movie_list = df_movie_summary[df_movie_summary['count'] < movie_benchmark].index
```

```
print('Movie minimum times of review: {}'.format(movie_benchmark))

df_cust_summary = df.groupby('user')['rating'].agg(f)
df_cust_summary.index = df_cust_summary.index.map(int)
cust_benchmark = round(df_cust_summary['count'].quantile(0.7),0)
drop_cust_list = df_cust_summary[df_cust_summary['count'] < cust_benchmark].index

print('Customer minimum times of review: {}'.format(cust_benchmark))
```

```
# Exclude movies, and users with less reviews
print('Original Shape: {}'.format(df.shape))
df = df[~df['movie'].isin(drop_movie_list)]
df = df[~df['user'].isin(drop_cust_list)]
print('After Trim Shape: {}'.format(df.shape))
print('-Data Examples-')
print(df.iloc[::5000000, :])
```

```
df.to_csv('data_trimmed.csv', index = None, header=True)
```

```
df_rates = pd.read_csv('data_trimmed.csv', parse_dates=['date'])
```

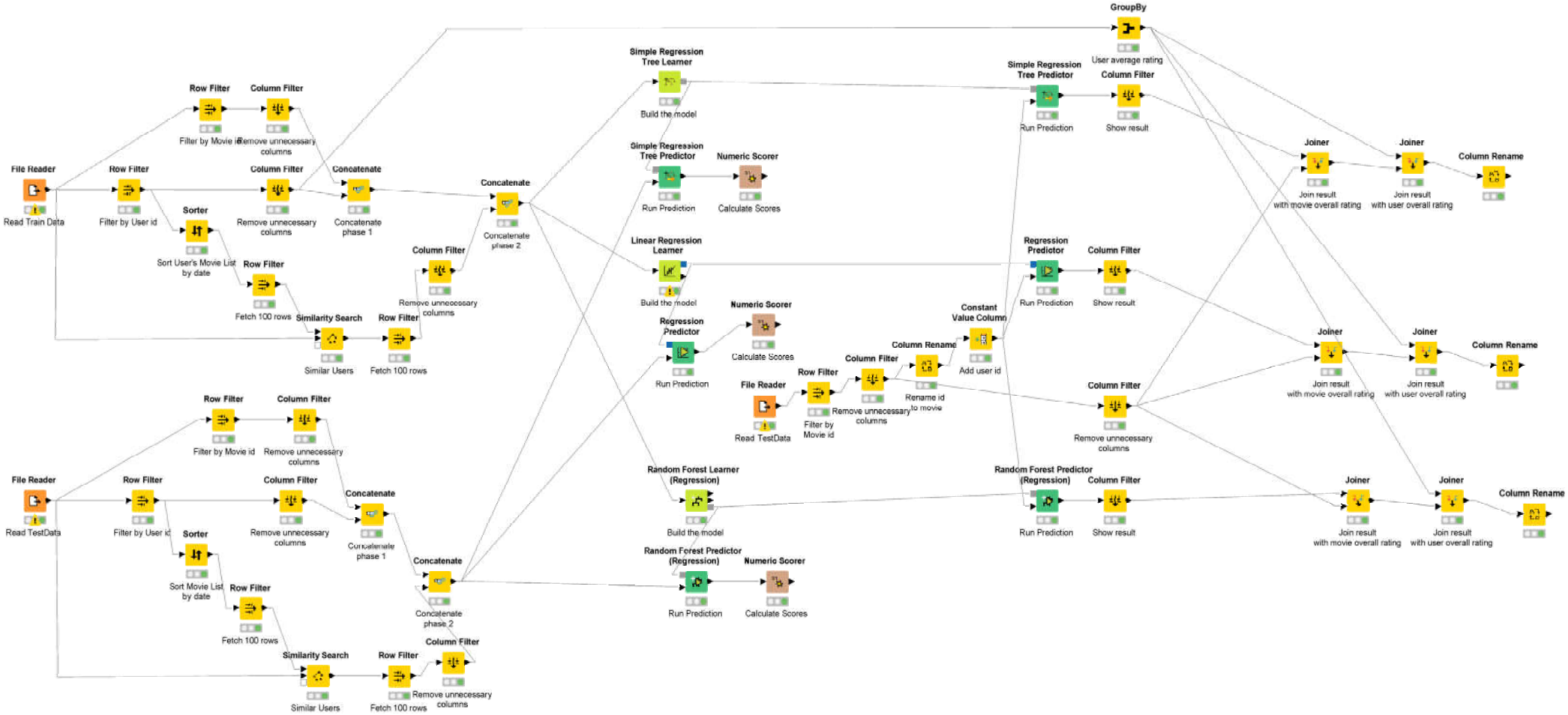
```

```

```
# Process movies
df2 = pd.read_csv('input/cooked_movies.csv')
meanVal = df2['Runtime'].mean()
df2['Runtime'].fillna(round(meanVal), inplace=True)
df2['Genres'].fillna('Other', inplace=True)
meanVal = df2['Rating'].mean()
df2['Rating'].fillna(round(meanVal,1), inplace=True)
df2['Directors'].fillna('Other', inplace=True)
df2['Writers'].fillna('Other', inplace=True)
df2['Production companies'].fillna('Other', inplace=True)
df2['year'].fillna(2000.0, inplace=True)
df2['Genres'] = df2['Genres'].str.replace('Sci-Fi','SciFi')
df2['Genres'] = df2['Genres'].apply(lambda x : x.split('-'))
df2.to_csv('movies_fixed.csv', index = None, header=True)
df_movies = pd.read_csv('movies_fixed.csv')

# Apply encoding to genres
from sklearn.preprocessing import MultiLabelBinarizer
mlb = MultiLabelBinarizer()
df_genres_mlb = pd.DataFrame(mlb.fit_transform(df_movies['Genres']),columns=mlb.classes_)
df_movies = pd.concat([df_movies, df_genres_mlb], axis=1)
df_movies = df_movies.drop(['Genres', 'Writers', 'Production companies', 'Directors', 'title'], axis=1)

# Merge rating with movies
df_rates_detailed = pd.merge(df_rates, df_movies,  how='left', left_on='movie', right_on='id')
df_rates_detailed = df_rates_detailed.drop(['id'], axis=1)
df_rates_detailed.to_csv('rates_movies_joined.csv', index = None, header=True)
```


File Reader
Read Train Data
Row Filter
Filter by User id
Sorter
Sort User's Movie List by date
Row Filter
Fetch 100 rows
Similarity Search
Similar Users
Row Filter
Fetch 100 rows
Column Filter
Remove unnecessary columns
Row Filter
Filter by Movie id
Column Filter
Remove unnecessary columns
Concatenate
Concatenate phase 1
Concatenate
Concatenate phase 2
File Reader
Read TestData
Row Filter
Filter by User id
Sorter
Sort Movie List by date
Row Filter
Fetch 100 rows
Similarity Search
Similar Users
Row Filter
Fetch 100 rows
Column Filter
Remove unnecessary columns
Row Filter
Filter by Movie id
Column Filter
Remove unnecessary columns
Concatenate
Concatenate phase 1
Concatenate
Concatenate phase 2
Simple Regression Tree Learner
Build the model
Simple Regression Tree Predictor
Run Prediction
Numeric Scorer
Calculate Scores
Linear Regression Learner
Build the model
Regression Predictor
Run Prediction
Numeric Scorer
Calculate Scores
Random Forest Learner (Regression)
Build the model
Random Forest Predictor (Regression)
Run Prediction
Numeric Scorer
Calculate Scores
File Reader
Read TestData
Row Filter
Filter by Movie id
Column Filter
Remove unnecessary columns
Column Rename
Rename id to movie
Constant Value Column
Add user id
GroupBy
User average rating
Simple Regression Tree Predictor
Run Prediction
Column Filter
Show result
Regression Predictor
Run Prediction
Column Filter
Show result
Random Forest Predictor (Regression)
Run Prediction
Column Filter
Show result
Column Filter
Remove unnecessary columns
Joiner
Join result with movie overall rating
Joiner
Join result with user overall rating
Column Rename
Joiner
Join result with movie overall rating
Joiner
Join result with user overall rating
Column Rename
Joiner
Join result with movie overall rating
Joiner
Join result with user overall rating
Column Rename