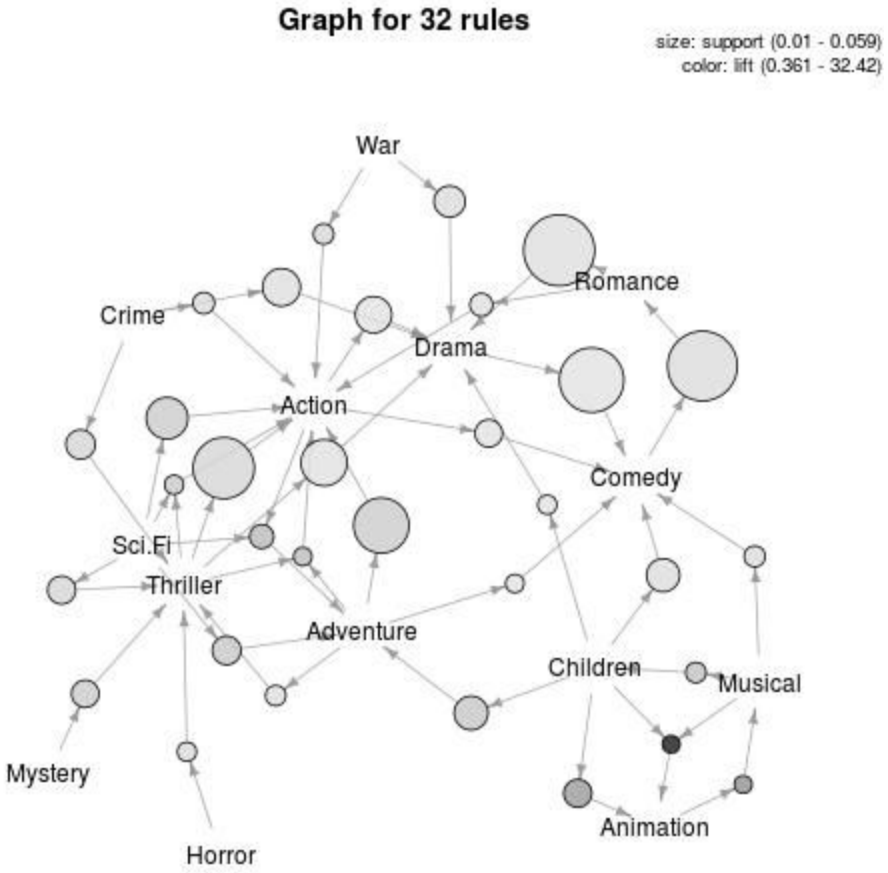
Финален проект по предметот
Податочно Рударење

Обработка на податоци поврзани со филмови, превеземени од Movie Lens податочно множество



Изработил: Горан Илиев, индекс 121145

За податоците

Податоците се преземени од следниов линк:
http://grouplens.org/datasets/movielens/.

Истите се однесуваат на филмови и оценки дадени од различни корисници за филмови. Множеството е прилично големо и содржи 100,000 оценки дадени од вкупно 943 корисници на 1682 различни филмови.

За релизација на барањата во вежбите го користам R програмскиот јазик, а за одредени задачи имам работено и во Python. Програмските кодови, како и податочните множества може да се видат на мојот github профил на следниов линк: https://github.com/goraniliev/movieLens

Користам R 3.2.2 и Python 2.7, значи на овие верзии кодовите проверив дека работат. Со правилно подесување на патеките до фајловите (на пример во R каде што има setwd() и getwd()) треба да работат сите кодови без проблем.

Изворните кодови во R се достапни на:
https://github.com/goraniliev/movieLens/tree/master/R

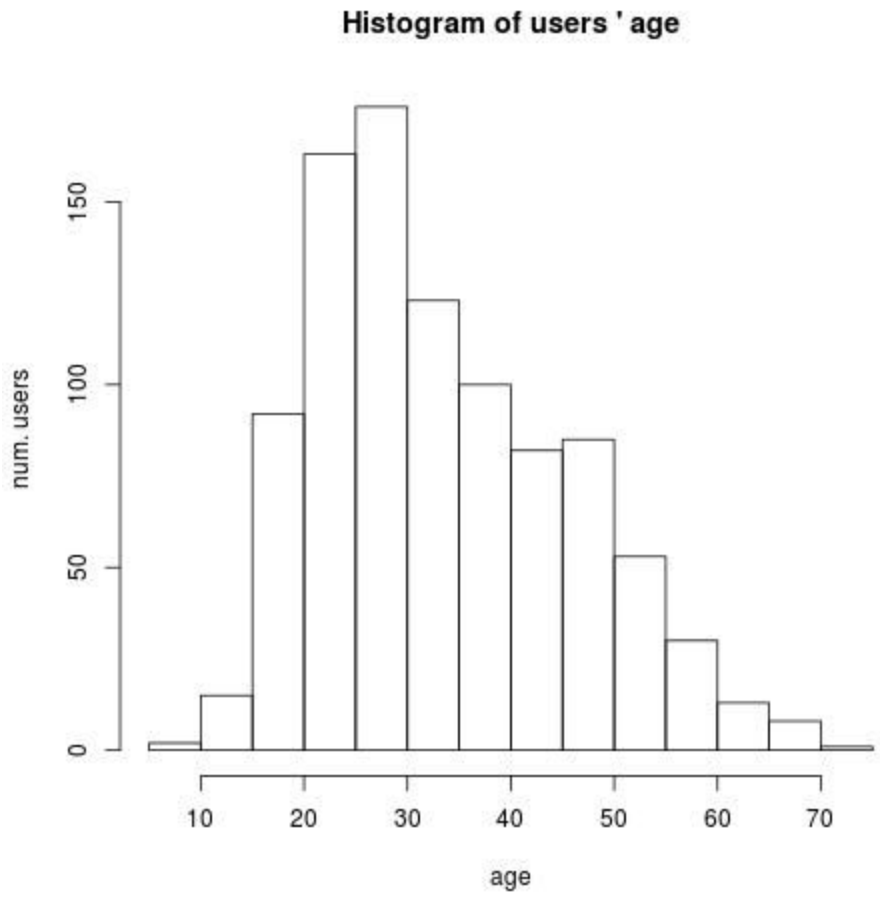
Резултатите добиени од пресметките ги зачував во текстуални фајлови и слики кои се достапни на github, на следниов линк: out.

1. Препроцесирање

Код: https://github.com/goraniliev/movieLens/blob/master/R/preprocessing.R

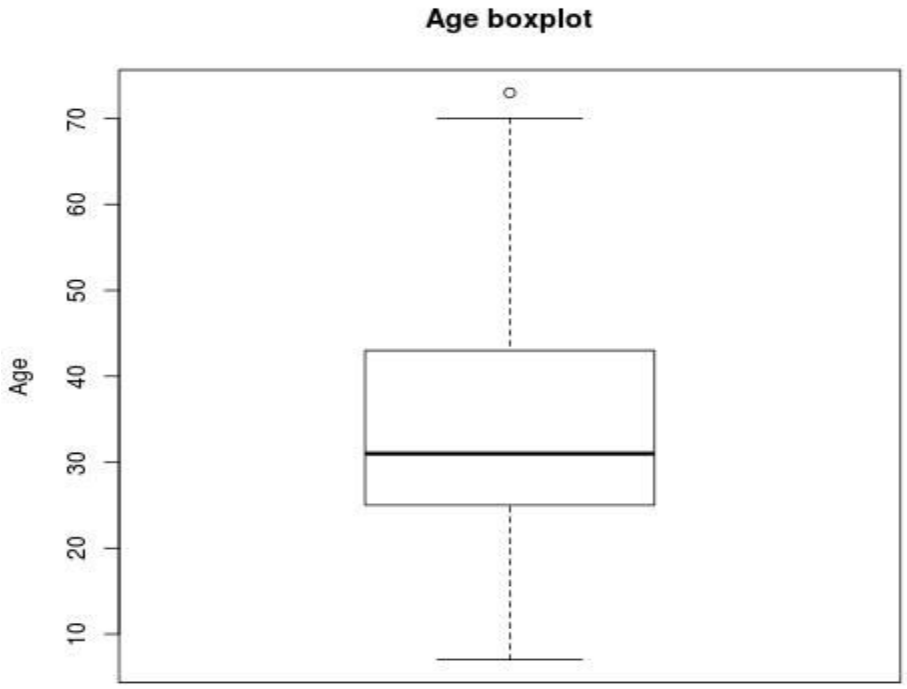
Во preprocessing.R вршам препроцесирање и разгледувам некои основни статистики за податоците. Резултатите ги има на github, па овде само ќе издвојам неколку.

Распределбата на возраста на корисниците (без нормализација) прикажана со хистограм:



Се забележува дека најмногу корисници има на возраст од 20 до 30 години. Исто така се забележува дека многу е мал бројот на корисници помлади од 15 или постари од 60 години.

Распределбата на возраста може да се прикаже и со boxplot, како на следниов boxplot:



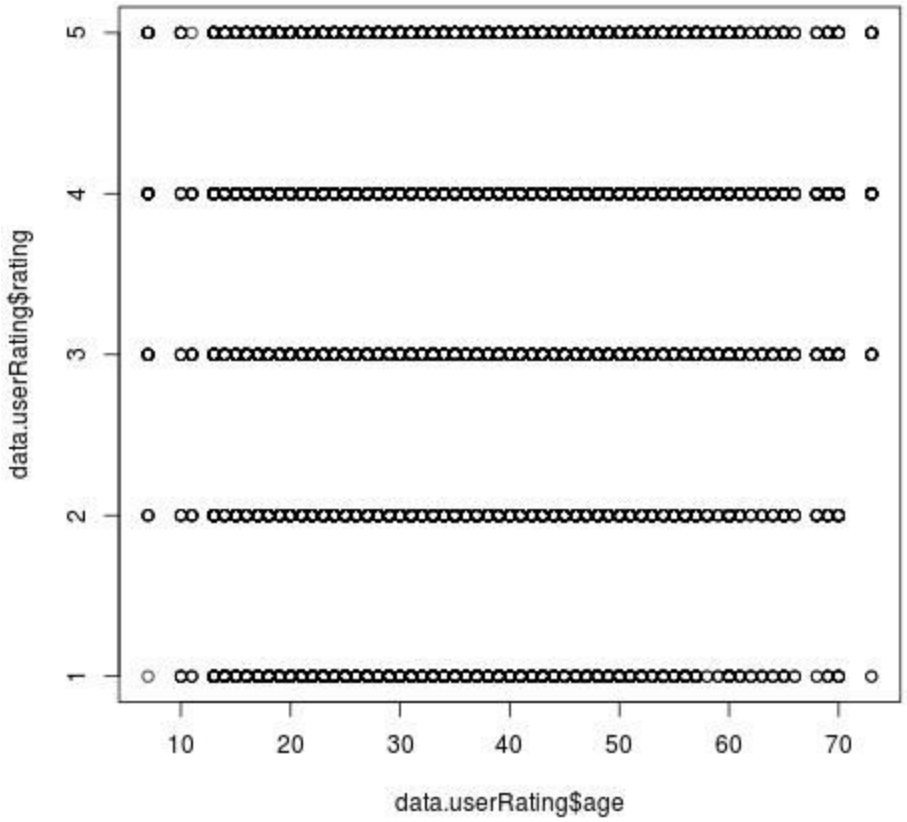
И овде се забележува дека медијаната е кај возраст од 30 години. Првиот и третиот квартил, се (приближно) кај возраст од 25 и 43 соодветно. Тоа значи дека 50% од луѓето се на возраст меѓу 25 и 43 години. Освен тоа, од графикот може да се види и дека границите за outliers се 7 и 70 приближно. Се забележува дека има само 1 outlier, во горниот дел од сликата, со возраст поголема од 70 години.
Во графички прикажаните резултати може да се увериме ако го погледнеме резултатот од summary() функцијата во R:

```
      age              gender        occupation
 Min.   : 7.00        F:273        student      :196
 1st Qu.:25.00        M:670        other        :105
 Median :31.00                     educator     : 95
 Mean   :34.05                     administrator: 79
 3rd Qu.:43.00                     engineer     : 67
Max.   :73.00                      programmer   : 66
                                   (Other)      :335
```

Освен возраста, овде е прикажана распределаба и на категориските атрибути, возраст и професија. Сето ова може подобро да се разгледа во фајловите preprocessing.txt и preprocessingNormalized.txt.

Слични вакви графици имам и за распределбата на оценките кои корисниците ги давале на филмовите. Исто така, имам креирано вакви графици и после нормализација на податоците. Истите може да се видат на github, но овде се одлучив да ги поставам ненормализираните бидејќи имаат поголемо значење и се поинтересни за човек.

Покрај histogram и boxplot, направив и scatter plot. Очекувано, меѓу возраста и рејтингот немаше зависност, бидејќи секој корисник има давано голем број рејтинзи и различни, па затоа нема некоја силна корелираност. Во прилог е сликата:



2. Класификација со Баесов класификатор и K најблиски соседи

2.1 Баесов класификатор

Код: https://github.com/goraniliev/movieLens/blob/master/R/bayes.R

Во оваа, како и во следните скрипти ги користам податоците добиени од preprocessing.R, па најпрво ја повикувам оваа скрипта.

Потоа, ги отфрлам незначајните атрибути, со тоа што креирам data.table modelData во која се содржат само значајните податоци за Баесовиот класификатор (имињата и id вредностите на филмовите ги отфрлив, бидејќи не се значајни за класификацијата).

Потоа вршам поделба на податоците во data_train (80%) и data_test (20%), со повикување на следниве наредби (createDataPartition е овозможен со библиотеката caret):

```
splitIndex = createDataPartition(model.data$rating, p=0.8, list=FALSE)
data_train = model.data[splitIndex[,1],]
data_test = model.data[-splitIndex[,1],]
```

И потоа го тренирам моделот со користење на метод од klaR пакетот:

```
model = NaiveBayes(as.factor(rating) ~., data=data_train)
```

Правам предвидувања:

```
predictions = predict(model, data_test[,-length(data_test), with = F])
```

И на крај сумаризација:

```
summarization = confusionMatrix(predictions$class, data_test$rating)
```

Сумарните резултати ги зачував во фајлот summarization.txt.

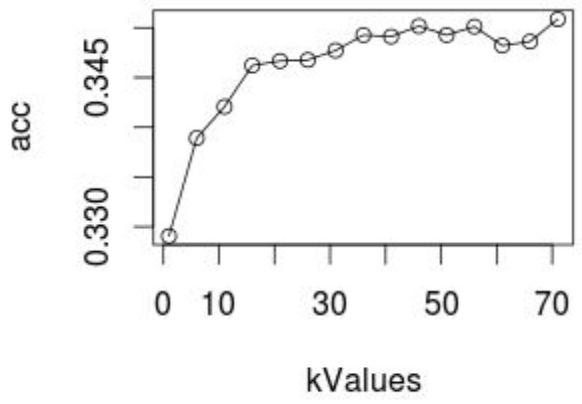
Прецизноста изнесува 0.3034, но тоа и не е така лошо ако се земе во предвид дека при погодување на случаен начин точноста би била 0.2 = ⅕ (бидејќи има вкупно 5 можни оценки за филм). Тоа е исто како во случај на бинарна класификација да се добие 0.75, што и не е многу лоша прецизност.

2.2 KNN класификатор

За креирање на модел за предвидување на оценки со KNN класификатор користам сличен пристап како кај Наивен Баесов класификатор. Креирам data table која ќе ги содржи податоците потребни за моделот, ги делам тие податоци на тренинг и тест множество. Овде, битно е да се напомене дека сите податоци треба да бидат нумерички. За професиите на корисниците не најдов соодветен начин да ги мапирам во нумерички вредности, па затоа не ги користам. Податоците во пол, бидејќи се бинарни вредности F и M ги мапирав во вредности 1 и 2 и ги користам истите.

За да немаат различни атрибути различно влијание извршив нормализација. Имено, доколку возраста ја оставев да има опсег на вредности од 7 до 77, возраста ќе имаше по големо влијание на предвидувањата. Затоа ги нормализирав овие вредности во интервал [0, 1].

За избор на оптималната вредност за k, креирав различни модели со различни вредности за k.



Како оптимална вредност ја одбрав k = 46. Se забележува дека за k = 71 се добива подобра прецизност, но за помало k побргу се врши предвидувањето, а разликата во прецизност не е голема, па затоа се одлучив за k = 46.

На крај, сумаризацијата за креираниот модел со одбрано k = 46 ја запишувам во фајл knn summarization.

Споредба на резултатите добиени од Наивен Баесов Класификатор и KNN класификатор

Во 13-ти ред од посоченият фајл може да се види дека прецизноста на моделот изнесува 0.3519, што е подобра прецизност споредено со 0.3034 за Наивниот Баесов Класификатор. Една од причините е можеби тоа што KNN подобро се справува со нумерички податоци, па возраста веројатно овде ја подобрува класификацијата. Исто така, Наивниот Баесов претпоставува условна независност меѓу атрибутите, а очигледно во конкретново податочно множество има зависност (на пример меѓу различните жанрови). Значи во конкретниов случај, KNN е за нијанса по прецизен од Наивниот Баесов класификатор. Но Наивниот Баесов има една огромна предност, тој еднаш се креира и потоа брзо ќе класифицира инстанци (т.н. eager learning). За разлика од него, KNN моделот класификацијата ја врши со интензивни пресметки во моментот на пристигнување на инстанците кои треба да се класифицираат (lazy learning), па истиот одзема многу повеќе време. За практични апликации можеби Naive Bayes би бил подобар избор, бидејќи е доста побрз, а разликата во прецизност не е значајна. Во случај на категориски атрибути и независност меѓу атрибутите Naive Bayes би бил супериорен. Во случај на нумерички атрибути (и силна зависност меѓу атрибути), подобри резултати би дал KNN.

3. Класификација со дрва и правила

3.1 Дрво на одлука

Код: https://github.com/goraniliev/movieLens/blob/master/R/bayes.R

Го користев ctree од party пакетот. Првин ги отфрлам незначајните атрибути како имиња и id. Потоа вршам поделба на множеството: 80% тренинг и 20% тест. Го креирам моделот и на крај вршам сумаризација на добиените резултати во фајлот Decision Tree summarization. Може да се забележи во контингенциската табела дека најголемите вредности се во главната дијагонала (или блиску до неа) и колку подалеку е некое поле од главната дијагонала толку е помала вредноста. Ова покажува дека дрвото на одлука најчесто дава вредности блиску до точните вредности. Може да се види на 16 ред од наведениот фајл дека прецизноста е 0.364, значи прецизноста добиена со дрво на одлука е подобра и од Наивен Баесов и од KNN класификаторот. Предностите кои ги има овој начин на класификација се: Во однос на Наивен Баесов - не претпоставува условна независност и нема проблем со зависни атрибути. Во однос на KNN - работи и со категориски атрибути и е eager learning метод на класификација, што значи многу побргу може да ја врши класификацијата на новодобиена инстанца. Недостатоци на овој пристап се тоа што може бргу да стане моделот доста комплексен и тоа да предизвика побавно одлучување, но што е уште полошо тешко ќе се визуелизира и разбира работата на ваквиот модел. Исто така, Дрвото на одлука не генерализира многу добро и можно е да има т.н. overfitting.

Немам поставено визуелен приказ од дрвото на одлука, бидејќи истото е премногу големо и не може да се претстави добро (има премногу преклопувања на јазли и ништо не може да се прочита и види јасно).

3.2 Правила

Немам имплементирано модел за одлучување со користење на правила бидејќи тој ќе биде со доста слична ефикасност и прецизност како и дрвотоа на одлука, затоа што во најголемиот број пакети за креирање на правила, истите се креираат од веќе креирано дрво на одлука. Затоа во продолжение имам креирано уште 2 модели за класификација - Random Forest и Логистичка Регресија.

3.3 Random Forest

Код: https://github.com/goraniliev/movieLens/blob/master/R/randomForest.R

Random Forest е техника за класификација која креира повеќе дрва на одлука и ги комбинира потоа за да добие попрецизни резултати во однос на користење на едно дрво на одлука. На тој начин, со градење на повеќе дрва и нивно комбинирање се избегнува потенцијален overfitting. Го користам пакетот randomForest од R.

Резултатите од предвидувањето се зачувани во Random Forest Summary. Прецизноста е 0.3686, за нијанса подобра од таа на Decision Tree.

3.4 Логистичка Регресија

Код: https://github.com/goraniliev/movieLens/blob/master/R/logitRegression.R

Логистичка регресија е доста често применувана техника за предвидување на класи кога имаме голем број на нумерички атрибути. Техниката работи слично како Линеарна Регресија, при што се става некоја прагова вредност и во зависност дали добиената вредност е над/под праговата вредност се одредува класата на која што припаѓа конкретната инстанца од податоци која што сакаме да ја класифицираме.

Го користев пакетот nnet во R и неговата функција multinom која креира мултиномна логистичка регересија со користење на невронска мрежа. Прецизноста на ваквата класификација е 0.344, а подетален приказ со сумаризација од ваквата класификација може да се најде на Logistic Regression Summarization.

Во делот 3, од тестираните различни техники за класификација најдобра прецизност даде Random Forest, па Decision Tree и на крај Логистичката Регресија. Како и да е, ваквото рангирање не мора да е секогаш исто, во зависност од податоците во различни ситуации може различен алгоритам да се покаже како најдобар.

4. Кластерриање и анализа на Асоцијации

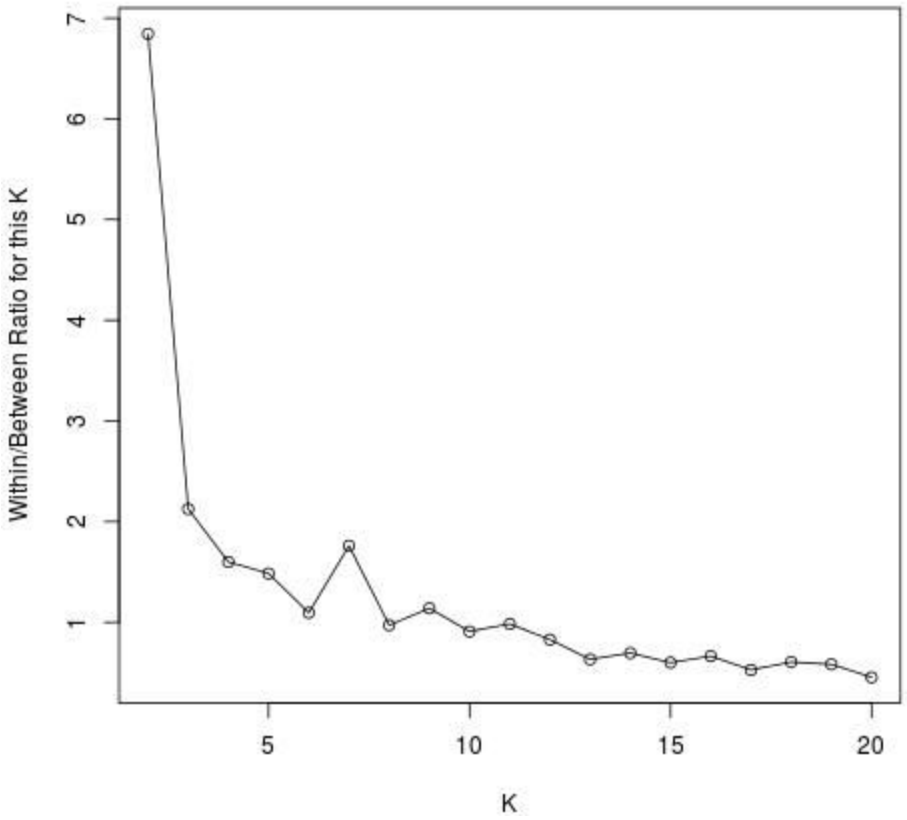
Во овој дел имам користено K-means, Hierarchical Clustering и Анализа на Асоцијации. Податоците ги земам од itemsData.txt, фајл кој го генерирав со Python скрипта. Имено, во тој фајл ги зачувувам само податоците потребни за кластерирање, а ги отфрлам непотребните.

4.1 Кластерирање со K-means

Код: https://github.com/goraniliev/movieLens/blob/master/R/Kmeans.R
Ги кластерирам филмовите според припадноста на жанрови, со цел на крај филмовите со слични жанрови да бидат во исти кластер, што практично би било доста корисно за препорака на филмови врз основа на веќе гледани филмови.

Ја користам функцијата kmeans од пакетот cluster во R. Истата ја повикувам за вредности за k во интервалот [2, 20]. Ги пресметав within / between односите за секое k (целта ни е да имаме кластер со помали растојанија внатре во кластерот (within) и поголеми растојанија меѓу одделните кластери (between), па затоа целта ни е да одбереме k кое ќе дава мала вредност за within / between). Како и да е, генерално гледано, овој однос монотоно ќе опаѓа со зголемување на k, но ние ќе избереме вредност за k после која со зголемување на k нема да добиеме некое големо подобрување на односот.

После пресметувањето на овие односи, го исцртав графикот (следна страна) и воочив дека k = 10 е добар кандидат за број на кластери. Може да се воочи дека и k = 6 дава приближно еднаков однос. Не одбрав k = 6, бидејќи во вредностите во негова непосредна близина (5 и 7) се доста поголеми, па можеби оваа вредност е добиена случајно и во друго стартување на kmeans можеби ќе биде доста поголема. Во околината на k = 10 и другите вредности се помали, а и знаејќи го значењетот на моето кластерирање (кластерирање на филмови според жанрови), претпоставив дека кластерирање со 10 кластери ќе даде подобри резултати.



Во k-means stats може да се видат генерални информации за кластерирањето. На пример, големината на секој од 10-те кластер е:

95 336 600 181 77 46 65 66 83 133

Сумата на квадратите од растојанијата внатре во кластерот е
within.cluster.ss = 224.7685.

Останатите детали може да се видат во k-means stats.

Во k-means clusters може да се видат кластерите. За секој кластер прво печатам Cluster [idCluster] и потоа во нов ред името на секој од филмовите.

За да проверам, направив проверки на IMDB. Во k-means clusters пребарав за филмот Forrest Gump и проверив неколку филмови кои се во истиот кластер (тоа е кластер број 2).

Во прилог се неколку филмови од кластерот и нивните оценки на IMDB:

Forrest Gump:	Drama, Romance
While You Were Sleeping:	Comedy, Drama, Romance
Ace Ventura: Pet Detective:	Comedy
Shall We Dance?:	Comedy, Drama, Romance
Cold Comfort Farm:	Comedy, Romance

Најверојатно ова е кластер на филмови кои припаѓаат комедија, драма, романса и некои сродни жанрови. Слична рачна анализа направив уште на неколку кластери и забележав дека слични типови на филмови се кластерирани во исти кластер, па може да се заклучи дека кластерирањето навистина дава добри резултати.

4.2 Хиерархиско кластерирање

Код: https://github.com/goraniliev/movieLens/blob/master/R/HierarchicalClustering.R

За Хиерархиско Кластерирање повторно го користам cluster пакетот од R. Сега ја користам функцијата hclust. Повторно бројот на кластери го одбирам на истиот начин, со тестирање на вредности за k од [2, 20]. За Хиерархиско кластерирање може да се одреди бројот на кластер и така што ќе се врши спојување на кластери се додека има кластери со растојание помало од некоја предефинира вредност, но во функцијата која ја користам не најдов таква опција, па го користам овој начин, кој на крај се покажа дека работи доста добро и за Хиерархиско Кластерирање.

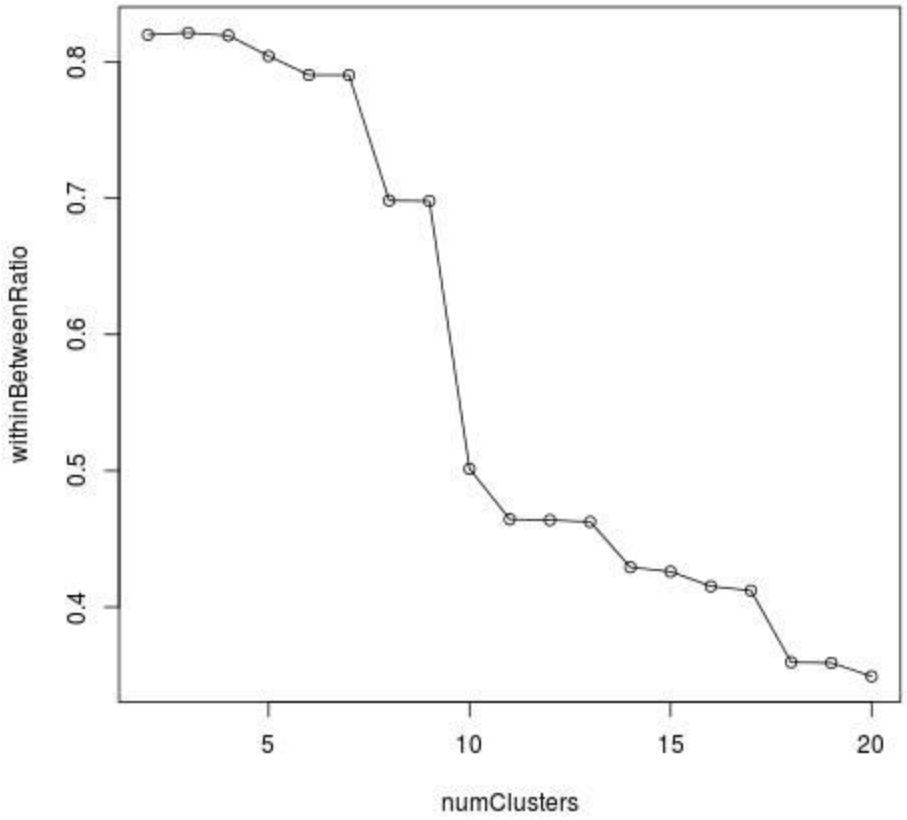
Применувајќи го истиот пристап како кај K-means, овде анализирајќи го графикот од следната слика, кој ги прикажува различните вредности за соодносот within/between како функција од бројот на кластери, одлучив да користам 11 кластери.

Потоа, вршам кластерирање на податоците со hcluster функцијата и правам пресек така што добивам 11 кластери.

Откако заврши кластерирањето, генералната статистика за кластерирањето ги запишав во hac stats. Може да се воочи од бројот на филмови во секој кластер:

```
482 216 144 606  48  10  80  69   9  16   2
```

дека ваквото кластерирање генерира по не изедначени кластери според бројот на елементи, т.е. некои кластери се значително поголеми од останатите.



Повторно направив мала анализа на добиените резултатие и ги проверив филмовите и на IMDB. Сега Forrest Gump е во кластер со многу помал број на филмови (Cluster 6 кој содржи вкупно 10 филма).

Еве мала споредба на жанровите на неколку филма:

Forrest Gump:	Drama, Romance
In Love and War	Biography, Drama, Romance
A Farewell to Arms	Drama, Romance, War

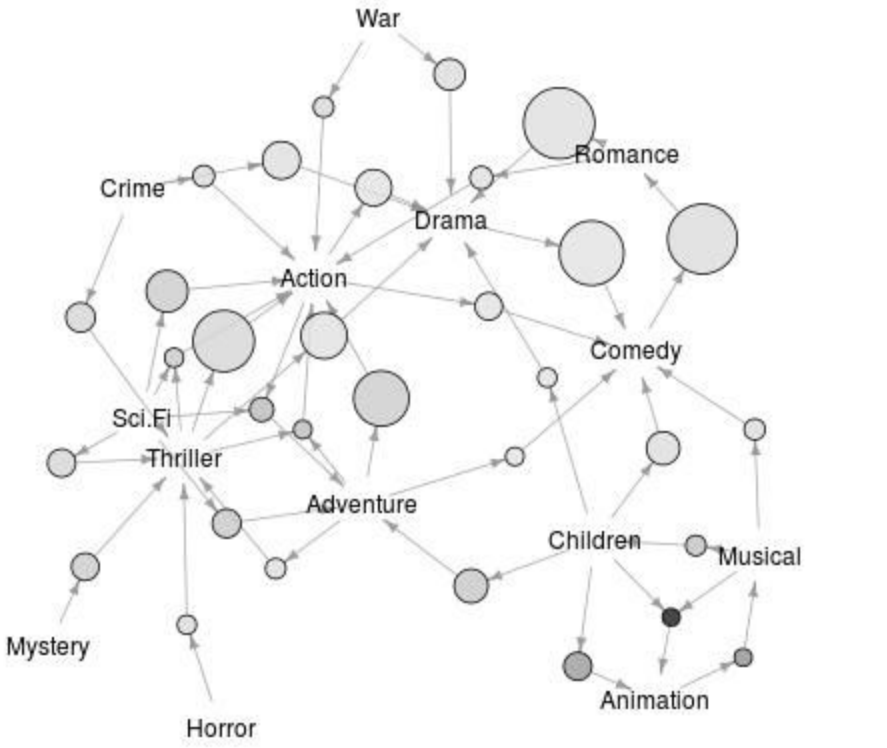
Се забележува дека и овде кластерирањето е прилично добро. Овде, за ралика од кај k-means, Forrest Gump е дел од многу помал кластер, бидејќи воочив дека филмовите кои припаѓаат на комедија не се вклучени овде, туку само тие кои се Драма, Романса, Биографски и сл. За таа сметка, има други кластери кои се доста поголеми.

Заклучокот е дека и двата алгоритми работат добро. K-means е значително побрз. Хиерархиското кластерирање е добро кога не сакаме да предвидуваме број на кластери, туку да комбинираме кластери се додека не се на растојание поголемо од дадена прагова вредност. Оваа предност на Хиерархиското Кластерирање не ја искористив, бидејќи во функцијата од R која ја користев не најдов таква опција.

4.3 Анализа на Асоцијации

Код: https://github.com/goraniliev/movieLens/blob/master/R/AssocitationRules.R
Го користев apriori алгоритамот од arules за креирање на асоцијациите и arulesViz за нивна визуелизација. Параметрите кои ги користам се minsup = 0.01, minconf = 0.1 и minlen = 2.

Првин ги генерирам сите правила, а потоа правам нивно поткастрување за да немам редундантни правила.
Сите генерирани правила (со редундантните) може да се видат во Redundant Association Rules.
Поткастрените правила (без редундантните) може да се видат во Pruned Rules.
Поткастрените правила се визуелизирани на следнава слика. Се гледа дека, на пример, филм кој е Мистерија со голема веројатност е и Трилер.


Graph for 32 rules
size: support (0.01 - 0.059)
color: lift (0.361 - 32.42)
War
Romance
Crime
Drama
Action
Comedy
Sci.Fi
Thriller
Adventure
Children
Musical
Mystery
Animation
Horror