УДК 004.9 https://doi.org/10.31713/vt2201827

Сопронюк Т. М., к.ф.-м.н., доцент (Чернівецький національний університет імені Юрія Федьковича, м. Чернівці)

ВІЗУАЛІЗАЦІЯ АЛГОРИТМУ FREQUENT PATTERN-GROWTH STRATEGY (FPG) ЗАСОБАМИ МОВИ R

У статті демонструється візуалізація алгоритму перетворення бази транзакцій у древовидну структуру, яка складається з дерева популярних предметних наборів (Frequent-Pattern Tree) й умовних дерев для окремих популярних предметів. Алгоритм FP-Growth зберігає в пам'яті компактну версію бази даних у вигляді дерева. Візуалізація FP-Growth Tree дерева та умовних дерев дає змогу наглядно продемонструвати переваги даного алгоритму. Візуалізація застосовується на прикладі аналізу ринкового кошика у супермаркеті. Такий аналіз є актуальним для розв'язання задач ефективної рекомендації (для перехресного продажу товарів).
Ключові слова: **FP-Growth, Data Mining, популярні предметні набори, рекомендації продуктів, умовне дерево, транзакція.**

Аналіз ринкового кошика (market basket analysis) включає в себе пошук найбільш частих типових, шаблонних покупок в супермаркетах (пошук асоціативних правил). Аналіз ринкового кошика проводиться для визначення комбінацій товарів, які пов'язані між собою. Це може бути корисним для рекомендації продуктів. Типові приклади - дані (набір продуктів) в супермаркеті або дані web-кліків. Компания Amazon, наприклад, використовує аналіз асоціативних правил для перехресного продажу товарів. Компания рекомендує клієнтам товари, грунтуючись на їх історії покупок та історії покупок інших клієнтів, які придбали той самий товар.

Основним алгоритмом, який застосовується для отримання асоціативних правил, є алгоритм Apriori (автор Rakesh Agrawal).

В R [1-4] існує пакет *arules* для побудови правил асоціації, Він використовує алгоритм Apriori. За допомогою розробки правил асоціації можна ідентифікувати елементи, які часто купуються разом.

У даній статті даному демонструється інший, більш ефективний, алгоритм FP-Growth [1-4]. Візуалізація відбувається засобами пакету *igraph* мови R.

У роботі [4] наведено ефективну реалізацію алгоритму FP-Growth мовою програмування C. Мова R, яка використовується в даній статті, має потужні можливості для візуалізації.

Алгоритм FP-Growth ефективно працює з будь-якою розподіленою системою, орієнтованої на mapReduce. FP-Growth розв'язує проблеми, що існують в Apriori, за допомогою спеціальної структури (дерева FP-Tree). У FP-Tree кожен вузол представляє елемент і його поточний лічильник, а кожна гілка представляє конкретну асоціацію.

Продемонструємо застосування алгоритму FP-Growth на прикладі аналізу кошиків у супермаркеті.

Основний алгоритм

Вхід: створення вхідного csv-файлу з транзакціями і завантаження його у пам'ять.

Крок 1: підрахунок всіх предметів у всіх транзакціях і сортування їх за спаданням частот.

Крок 2: фільтрування транзакцій (вибір транзакцій з потрібною підтримкою).

Крок 3: побудова дерева FP-Growth Tree.

Крок 4: побудова умовних дерев для кожного предмету.

Вихід: популярні набори предметів.

Візуалізація: дерева FP-Growth Tree і Conditional FP- trees.

Нехай у поточному каталозі ми зберігаємо файл з транзакціями *transactions.csv*.

Далі наведено фрагменти R-скрипта та візуалізація окремих структур даних.

```
>library(arules)
>library(igraph)
> file <- "transactions.csv"

## read transactions from csv-file
data <- strsplit(readLines(file), "[[:space:]]")
## create transaction data set
> tr <- as(data, "transactions")
> inspect(tr)
  items
[1] {juice,butter,cheese,milk,bread}
[2] {juice,butter,cheese,shampoo}
[3] {juice,cheese,milk,bread}
```

[4] {butter,cheese,milk,bread}
[5] {butter,cheese}
[6] {butter,milk,bread}
[7] {cheese,milk,bread}

Крок 1: підрахунок всіх предметів у всіх транзакціях і сортування їх за спаданням частот.

```
## read dataframe from csv-file
> df<-read.csv(file, head=F, stringsAsFactors=FALSE)
> freq<-rev(sort(table(unlist(df), exclude = "")))
> print(freq)
 cheese   milk  butter   bread  juice shampoo
 6    5      5     5      3    1
```

Крок 2: фільтрування транзакцій (приклад фільтру: кожен елемент має з'являтися щонайменше двічі).

```
>suffixes<-as.vector(as.data.frame(freq[freq>=2])[,1])
> print(suffixes)
 [1] "cheese" "milk"   "butter" "bread"  "juice"
```

Суфікс - це кінцевий вузол (в нашому випадку – продукт), до якого будується шлях в дереві. Таким чином, всі шляхи будуть мати різні префікси (комбінації продуктів) і однаковий суфікс.

Як бачимо, після фільтрування видалений продукт *shampoo*.

Для виконання наступних років алгоритму розроблено ряд функцій, які підключається до основної функції та використовуються в подальших розрахунках:

```
source('drawTree.R')
source('allBranches.R')
source('buildTree.R')
source('buildConditionalTree.R')
```

Крок 3: побудова дерева FP-Tree (рис. 1).

```
## Step 3: Build FP-Grofth tree
r<-buildTree(df,suffixes)
drawTree(r)
title("FP-GRowth Tree ")
```

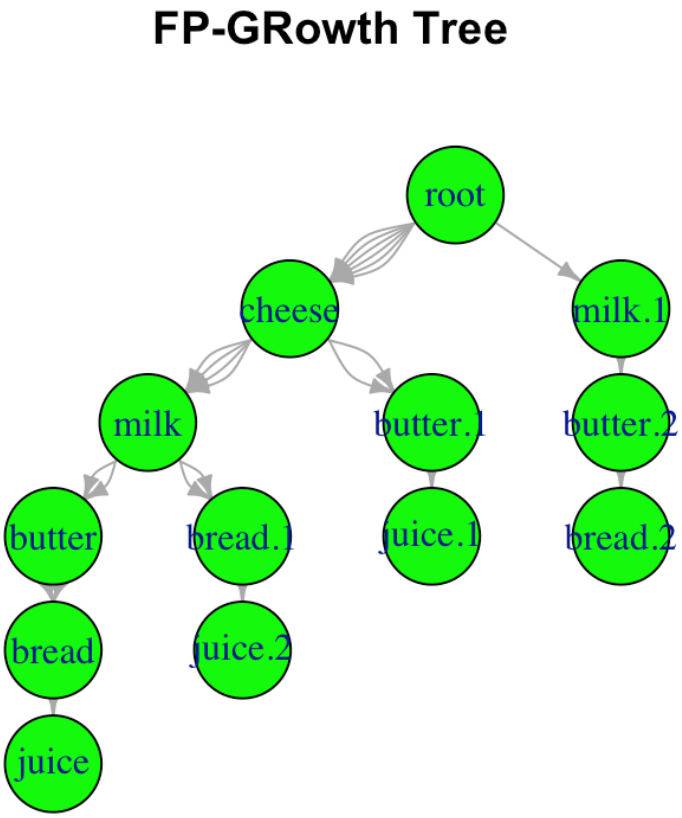

Рис. 1. Дерево FP-Tree

Дерево FP-Tree зберігає інформацію про частоту предметів у наборах.

Кількість ребер, направлених у вершину, навантажену назвою продукту, відповідає частоті продукту. Якщо для чергового предмета в дереві зустрічається вузол, ім'я якого збігається з іменем предмета, то кількість ребер, направлених у відповідний вузол, збільшується на 1. В іншому випадку для цього предмета створюється новий вузол і на нього вказує ребро. Вершини з однаковими префіксами в іменах вважаються ідентичними, наприклад, *bread*, *bread.1*, *bread.2*.

Візуалізація алгоритму побудови дерева FP-Tree відбувається за допомогою наведених нижче функцій.

```
## function for building FP-Growth tree
buildTree<-function(df,suffixes){
   n<-nrow(df)
   orderDf<-df
   el<-vector()
   roots <- vector(mode="list", length=n)
   for (i in 1:n){
      row<-as.vector(df[i,])
      v<-vector()
      for (j in 1:m){
         if(suffixes[j] %in% row)
            v<-c(v,suffixes[j])
      }
      for (j in 1:length(v))
         orderDf[i,j]<-v[j]

      v[] <- lapply(v, as.character)
      names(v)<-" "
      v<-rep(as.vector(v),each=2)
      v<-c("root",v)
      while(v[length(v)]=="")
         v<-v[1:length(v)- 1]
      roots[[i]]<-v[1:length(v)- 1]
   }

   roots<-allBranches(roots,n)
   #update df, according frequencies
   eval.parent(substitute(df<-orderDf))
   return (roots)
}
```

```
## function for drawing tree from lists
drawTree <- function(el) {
    matr <- matrix( as.character(unlist(el)), nc=2, byrow=TRUE)
    gr <- graph.edgelist(matr)
    co <- layout.reingold.tilford(gr, params=list(root=1))
    plot(gr, layout=co, vertex.size=34, vertex.color="green", edge.arrow.size=0.5)
}
```

```
## function for building all path for tree
allBranches <- function(r,size) {
    #create unique nodes, for example: bread, bread.1, bread
    setSymbol<-unique(unlist(r[[1]]))
    roots<-r
    if(size==1)
        return(roots)
```

```
  for (i in 2:size){
      max<-2
      kmax<-1
  for (k in 1:(i-1)){
      j<-2
      while (as.character(r[[k]][j]) == as.character(r[[i]][j]) &&
        j<=length(r[[k]]) && j<=length(r[[i]]))
      j<-j+1
      if (j>=max){
        kmax<-k
        max<-j
      }
  }
  c<-r[[i]][max]
  for (l in 1:(max-1))
      roots[[i]][l]<-roots[[kmax]][l]

  if(max > length(roots[[i]]))
      next
  if(max %% 2 >0 )
  {
      roots[[i]][max]<-roots[[kmax]][max-1]
      max<-max+1
  }
  while (max<=length(roots[[i]])) {
      c<-r[[i]][max]
      newSet <-unique(unlist(c(setSymbol, c)))
      if(length(setSymbol) != length(newSet)){
        setSymbol<-newSet
        if(max<length(roots[[i]])){
          max<-max+1
      }
  }
      else{
          newSet<-make.unique(unlist(c(setSymbol, c)))
          newEnge<-setdiff(newSet,setSymbol
          roots[[i]][max]<-newEnge
          setSymbol<-newSet
          if(max<length(roots[[i]])){
            max<-max+1
            roots[[i]][max]<-newEnge
          }
      }
      max<-max+1
  }
      }
      roots
}
```

Крок 4: побудова умовних дерев для кожного предмету.

Для кожного продукту в FP-дереві, можна вказати шлях (послідовність вершин від кореня до вузла, пов'язаного з даним продуктом). Шлях може складатися з декількох гілок. Кожен шлях складається з префіксу і суфіксу. Префікс – це послідовність вершин до потрібного продукту. А суфікс – це сам потрібний продукт. Умовне дерево представляє всі можливі префікси для потрібного продукту. Сам суфікс не є присутнім в умовному дереві.

Далі, використовуючи FP-дерево, будемо генерувати умовні дерева для продуктів, упорядкованих за спаданням їх популярності.

```
## Step 4: Frequent Itemset Generation
for (i in 1:(length(suffixes)-1)){
    item<-suffixes[length(suffixes)-i+1]
    r<-buildConditionalTree(df, item, suffixes)
    if(!is.null(r)){
        drawTree(r)
    }
    else{
        drawTree("root")
    }
    title(paste0("Conditional FP-tree for ",item))
}
```

Умовне дерево для продукту *juice* представлено на рис. 2.

Умовне дерево для продукту *bread* представлено на рис. 3.

Умовне дерево для продукту *butter* представлено на рис. 4.

Умовне дерево для продукту *milk* представлено на рис. 5.

Продукт *cheese* є безпосереднім нащадком кореневого вузла, тому не можна вказати ніяких шляхів й умовне дерево для цього продукту не будується. Отже, префікс шляхів для продукту *cheese* буде порожнім, а популярні предметні набори відсутні.

```
## function for building conditional tree
buildConditionalTree<-function(df, item, suffixes){
    dfe<-df
    v<-vector()
    for (i in 1:nrow(df)){
        row<-as.vector(df[i,])
        if(!(item %in% row)){
            v<-c(v,i)
        }
    }
    dfe<-df[-v,]
    for (i in 1:nrow(dfe))
    {
        row<-as.vector(dfe[i,])
        k<-length(row)+1

        for (j in 1:length(row))
            if(row[j]==item ){
              k<-j
              break
            }

    while (k<=length(row)){
        dfe[i,k]<-''
        k<-k+1
      }
    }

    emptycols <- colSums(dfe == "") == nrow(dfe)
    dfe <- dfe[!emptycols]
    if(ncol(dfe)==0)
        return (NULL)

    emptyrows <- rowSums(dfe == "") == (ncol(dfe))
    dfe <- dfe[!emptyrows,]
    dfe <- as.data.frame(dfe)

    if(nrow(dfe)==0)
        return (NULL)

    r<-buildTree(dfe, suffixes)
    return (r)
    }
```

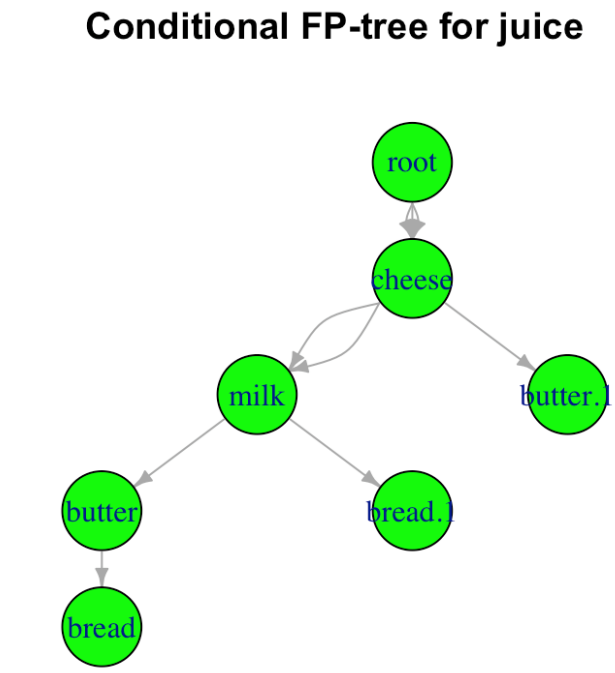

Рис. 2. Умовне дерево FP-Tree для продукту «сік»

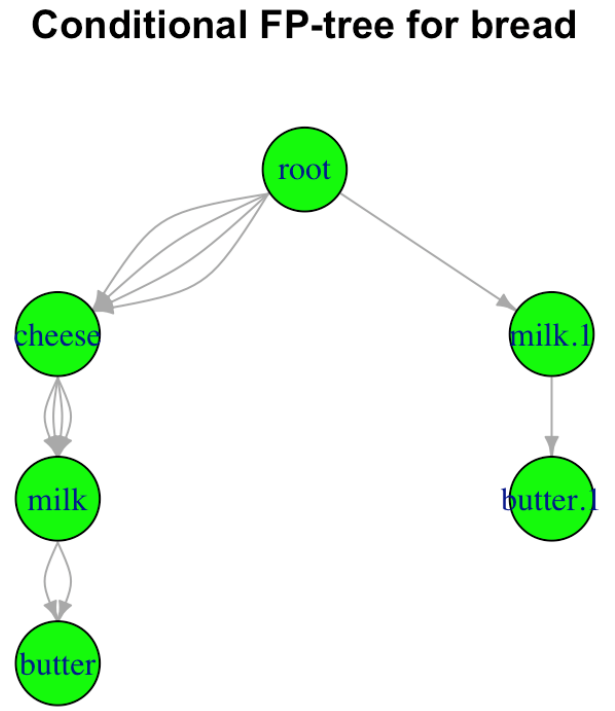

Рис. 3. Умовне дерево FP-Tree для продукту «хліб»

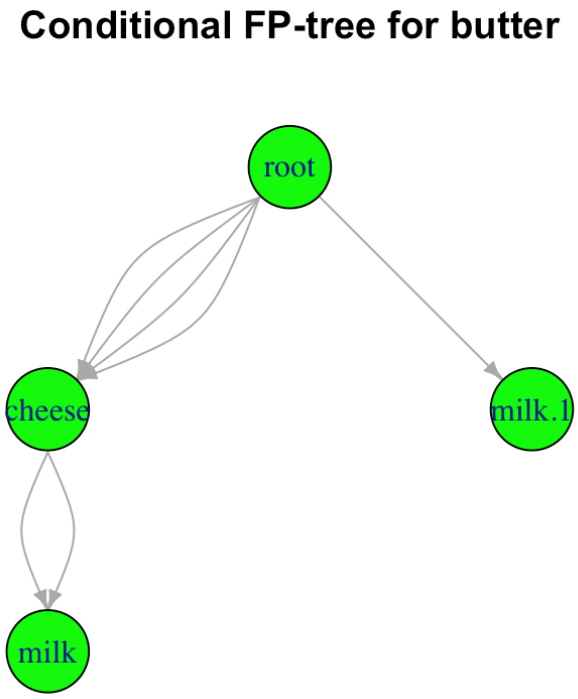

Рис. 4. Умовне дерево FP-Tree для продукту «масло»

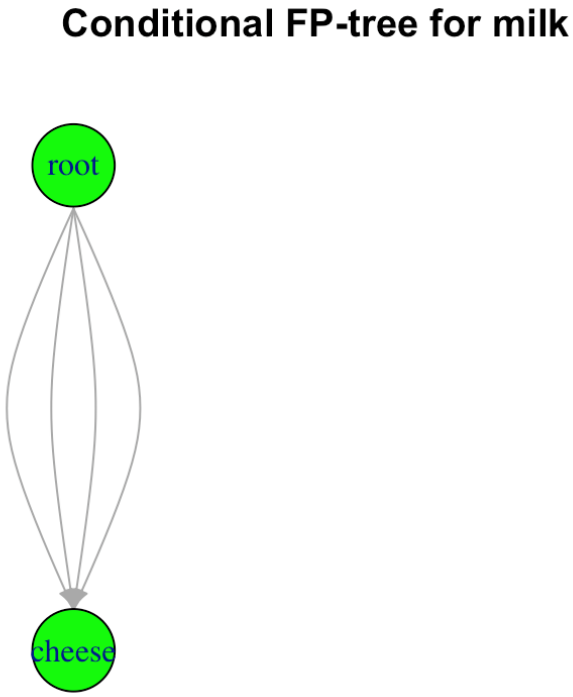

Рис. 5. Умовне дерево FP-Tree для продукту «молоко»

Об'єднуючи умовні дерева для кожного продукту, отримуємо результат роботи алгоритму (популярні набори предметів).

У першу чергу відкидаємо продукт s*hampoo,* оскільки його рівень підтримки дорівнює одиниці, тобто 14.2857%.

Рівень підтримки інших окремих продуктів нас влаштовує:

cheese (85.7143%)
butter (71.4286%)
milk (71.4286%)
bread (71.4286%)
juice (42.8571%)

Аналізуючи умовне дерево для продукту *juice* (рис. 2), отримуємо набір популярних продуктів з суфіксом *juice* (префікси відповідають окремим шляхам до продукту в умовному дереві):

cheese, juice (42.8571%)
cheese, milk, juice (28.5714%)
cheese, butter, juice (28.5714%)
cheese, bread, juice (28.5714%)
butter, juice (28.5714%)
bread, juice (28.5714%)
milk, juice (28.5714%)
milk, bread, juice(28.5714%)
cheese, bread, milk, juice (28.5714%)

Аналогічно, аналізуючи умовне дерево для продукту *butter* (рис. 3), отримуємо наступний набір продуктів:

cheese, butter (57.1429%)
milk, butter (42.8571%)
cheese, milk, butter (28.5714%)

Аналізуючи умовне дерево для продукту *bread* (рис. 4), отримуємо наступний набір продуктів:

milk, bread (71.4286%)
cheese, bread (57.1429%)
cheese, milk, bread (57.1429%)
butter, bread (42.8571%)

milk, butter, bread (42.8571%)
cheese, butter, bread (28.5714%)
cheese, milk, butter, bread (28.5714%)

Остаточно, з умовного дерева продукту *milk* (рис. 5), маємо набір продуктів:

cheese, milk (57.1429%)

У дужках наведений відсоток підтримки наборів відповідних продуктів.

Як бачимо, алгоритм FP-Growth зберігає в пам'яті компактну версію бази даних у вигляді дерева. Візуалізація FP-Growth Tree дерева та умовних дерев дає змогу наглядно продемонструвати переваги даного алгоритму. Він більш масштабний, ніж алгоритм Apriori.

1. *Pawel Cichosz*. Data Mining Algorithms: Explained Using R / Pawel Cichoz// John Wiley & Sons. 2015. 716 p. **2.** *Bater Makhabel*. Learning Data Mining with R / Bater Makhabel. // Packt Publishing Ltd. 2015. 314 p. **3.** *B. Santhosh.* Implementation of Web Usage Mining Using APRIORI and FP Growth Algorithms. / B. Santhosh Kumar, K. V. Rukmani // Int. J. of Advanced Networking and Applications. Volume: 01, Issue: 06. 2010. PP. 400–404. **4.** *Christian Borgelt*. An implementation of the fpgrowth algorithm. / Christian Borgelt. // In OSDM '05: Proceedings of the 1st international workshop on open source data mining. New York, NY, USA. ACM. 2005. PP. 1–5.

REFERENCES:

1. Pawel Cichosz. Data Mining Algorithms: Explained Using R / Pawel Cichoz // John Wiley & Sons. 2015. 716 p. **2.** Bater Makhabel. Learning Data Mining with R / Bater Makhabel. // Packt Publishing Ltd. 2015. 314 p. **3.** B. Santhosh. Implementation of Web Usage Mining Using APRIORI and FP Growth Algorithms. / B. Santhosh Kumar, K. V. Rukmani // Int. J. of Advanced Networking and Applications. Volume: 01, Issue: 06. 2010. PP. 400–404. **4.** Christian Borgelt. An implementation of the fpgrowth algorithm. / Christian Borgelt. // In OSDM 05: Proceedings of the 1st international workshop on open source data mining. New York, NY, USA. ACM. 2005. PP. 1–5.

Рецензент: д.т.н., професор Мартинюк П. М. (НУВГП)

Soproniuk T. M., Candidate of Physical and Mathematical Sciences (Ph.D.), Associate Professor (Yuriy Fedkovych Chernivtsi National University, Chernivtsi)

VISUALIZATION OF ALGORITHM FREQUENT PATTERN-GROWTH STRATEGY (FPG) IN R

Market basket analysis involves the search for the most common typical, popular shopping in supermarkets (search for associative rules). The analysis of the market basket is carried out to determine the combinations of items that are interconnected. Such an analysis is relevant for solving the problem of effective recommendation (for cross-selling goods). Using the results of the analysis, companies can recommend products to their customers, based on their purchasing history and purchasing history of other customers who purchased the same product.

This article demonstrates the algorithm for transforming the transaction database into a tree structure. This structure consists the Frequent-Pattern Tree and conditional trees for some popular items. Visualization is used for showing the analysis of a market basket in a supermarket and sold by means of the igraph package of R language.

Basic algorithm.

Input: csv-file with transactions.

Step 1: Count all items in all transactions and sort them down by frequency.

Step 2: Filter the transactions (select the transactions with the required support).

Step 3: Build a FP-Growth Tree.

Step 4: Construct conditional trees for each item.

Output: Popular sets of items.

Visualization: FP-Growth Tree and Conditional FP-trees.

***Keywords*: FP-Growth, Data Mining, Frequent Itemset Mining, popular objects, products recommendations, conditional tree, transaction.**

Сопронюк Т. Н., к.ф.-м.н., доцент (Черновицкий национальный университет имени Юрия Федьковича)

ВИЗУАЛИЗАЦИЯ АЛГОРИТМА FREQUENT PATTERN-GROWTH STRATEGY (FPG) СРЕДСТВАМИ ЯЗЫКА R

В статье демонстрируется визуализация алгоритма преобразования базы транзакций в древовидном структуру, состоящую из дерева популярных предметных наборов (Frequent-Pattern Tree) и условных деревьев для отдельных популярных предметов. Алгоритм FP-Growth сохраняет в памяти компактную версию базы данных в виде дерева. Визуализация FP-Growth Tree дерева и условных деревьев позволяет наглядно продемонстрировать преимущества данного метода. Визуализация применяется на примере анализа рыночной корзины в супермаркете. Такой анализ является актуальным для решения задач эффективной рекомендации (для перекрестных продаж товаров).
Ключевые слова: **FP-Growth, Data Mining, популярные предметные наборы, рекомендации продуктов, условное дерево, транзакция.**
