МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Факультет прикладной математики и информатики

Кафедра ТПИ

Дисциплина: «Основы теории машинного обучения»

Лабораторная работа №6

Вариант №1

**ПОИСК АССОЦИАЦИЙ**

Факультет: ФПМИ

Группа: ПМИМ-31

Выполнили: Монгуш Н. С., Тарулин М. А., Филипенко Ю. Д.

Проверил: Попов А. А.

Дата выполнения:

Отметка о защите:

1. ЦЕЛЬ РАБОТЫ

Получить практические навыки по выявлению в наборах данных ассоциативных правил.

1. ЗАДАНИЕ
2. Провести генерацию набора данных с транзакциями в соответствии с вариантом задания.
3. Реализовать алгоритм Apriori поиска ассоциативных правил.
4. Осуществить решение задачи поиска ассоциативных правил в раннее созданном наборе данных. Минимальный уровень поддержки задается пользователем и в процессе тестирования приложения варьируется. Оценить качество работы приложения.
5. ИСХОДНЫЕ ДАННЫЕ

Вариант задания:

Общее количество объектов 100. Общее число транзакций не менее 100000. При генерации набора данных использовать ассоциативные правила разного уровня поддержки. Число элементов в частных наборах используемых правил равно 2, 3, 4. Число правил с большим уровнем поддержки 3, со средним уровнем 5, с небольшим уровнем 10.

1. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Изобразим сгенерированные транзакции на рис. 1.1.

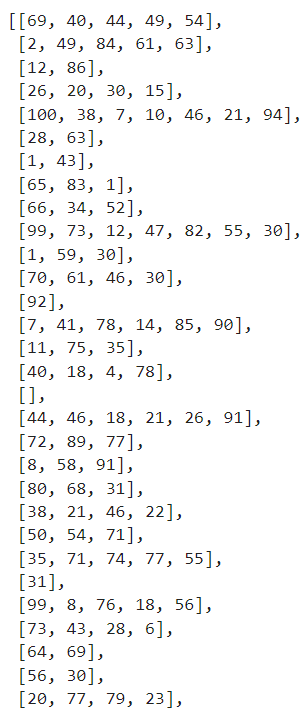


Рис. 1.1 – Сгенерированные транзакции.

Протестируем работу реализованного алгоритма Apriori с различными уровнями поддержки: с высоким уровнем поддержки – 10%, средним – 5%, низким – 1%. Результаты работы алгоритма изобразим на рисунках 2.1, 2.2 для каждого уровня соответственно. Для уровня 10% ассоциативные правила отсутствуют.

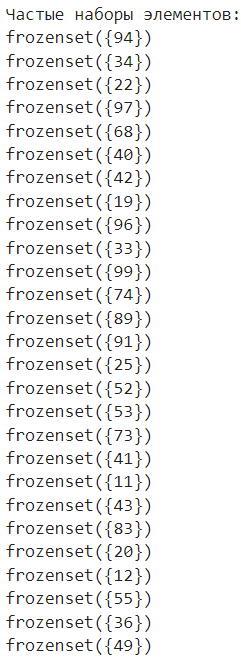
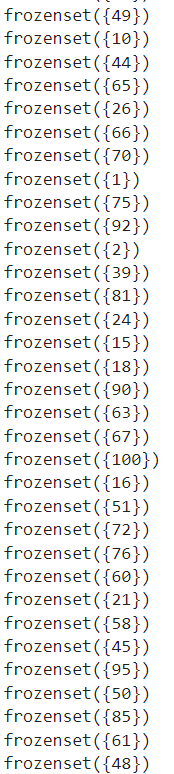
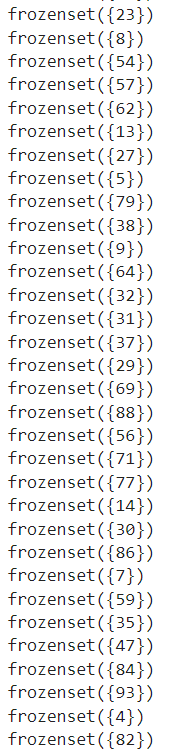
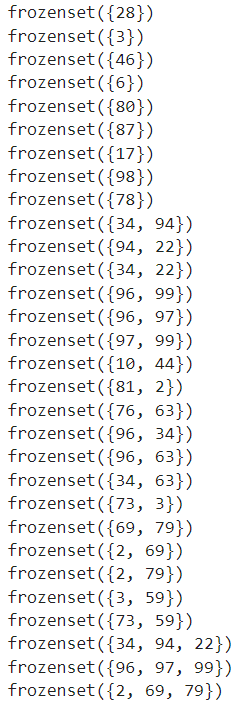
   

Рис. 2.1 – Результат поиска ассоциативных правил с 1% уровнем поддержки.

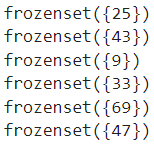


Рис. 2.2 – Результат поиска ассоциативных правил с 5% уровнем поддержки.

1. ВЫВОД

В процессе выполнения лабораторной работы были сгенерированы транзакционные данные в соответствии с заданным вариантом. Затем был реализован алгоритм Apriori для поиска ассоциативных правил в этих данных. После этого было проведено тестирование алгоритма с различными уровнями поддержки: высоким, средним и низким. Оказалось, что при уровне поддержки 10% ассоциативные правила отсутствовали, что указывает на высокую специфичность частых наборов при таком пороге. Полученные результаты были визуализированы на графиках.

1. КОД ПРОГРАММЫ

import random

import pandas as pd

# Константы для генерации данных

NUM\_ITEMS = 100

NUM\_TRANSACTIONS = 100000

NUM\_RULES\_HIGH\_SUPPORT = 3

NUM\_RULES\_MEDIUM\_SUPPORT = 5

NUM\_RULES\_LOW\_SUPPORT = 10

# Вспомогательная функция для генерации правил

def generate\_rules(num\_rules, itemset\_size):

rules = []

for \_ in range(num\_rules):

items = random.sample(range(1, NUM\_ITEMS + 1), itemset\_size)

rules.append(items)

return rules

# Генерация правил разного уровня поддержки

high\_support\_rules = generate\_rules(NUM\_RULES\_HIGH\_SUPPORT, 2)

medium\_support\_rules = generate\_rules(NUM\_RULES\_MEDIUM\_SUPPORT, 3)

low\_support\_rules = generate\_rules(NUM\_RULES\_LOW\_SUPPORT, 4)

# Вспомогательная функция для создания транзакции

def create\_transaction():

transaction = set()

# Добавление правил с вероятностью, соответствующей уровню поддержки

if random.random() < 0.1: # Высокая поддержка

transaction.update(random.choice(high\_support\_rules))

if random.random() < 0.05: # Средняя поддержка

transaction.update(random.choice(medium\_support\_rules))

if random.random() < 0.01: # Низкая поддержка

transaction.update(random.choice(low\_support\_rules))

# Добавление случайных элементов

num\_additional\_items = random.randint(0, 5) # До 5 дополнительных элементов

additional\_items = random.sample(range(1, NUM\_ITEMS + 1), num\_additional\_items)

transaction.update(additional\_items)

return list(transaction)

# Генерация транзакций

transactions = [create\_transaction() for \_ in range(NUM\_TRANSACTIONS)]

def generate\_candidates(prev\_itemsets, k):

"""Генерация кандидатов для нового уровня из предыдущего набора частых наборов"""

candidates = set()

for a in prev\_itemsets:

for b in prev\_itemsets:

union = a.union(b)

if len(union) == k:

candidates.add(union)

return candidates

def apriori(transactions, min\_support):

"""Реализация алгоритма Apriori для нахождения частых наборов"""

# Первый шаг: подсчёт одноэлементных частых наборов

item\_support = {}

for transaction in transactions:

for item in transaction:

itemset = frozenset([item])

if itemset in item\_support:

item\_support[itemset] += 1

else:

item\_support[itemset] = 1

n\_transactions = len(transactions)

itemsets = [itemset for itemset, count in item\_support.items() if count / n\_transactions >= min\_support]

# Хранение всех частых наборов

frequent\_itemsets = list(itemsets)

k = 2

# Итерация по уровням

while itemsets:

candidates = generate\_candidates(itemsets, k)

support\_counts = defaultdict(int)

for transaction in transactions:

for candidate in candidates:

if candidate.issubset(transaction):

support\_counts[candidate] += 1

# Фильтрация кандидатов с достаточной поддержкой

itemsets = [itemset for itemset, count in support\_counts.items() if count / n\_transactions >= min\_support]

frequent\_itemsets.extend(itemsets)

k += 1

return frequent\_itemsets

# Пример использования

#transactions = load\_transactions() # Загрузка транзакций из файла

min\_support = 0.1 # Минимальный уровень поддержки

frequent\_itemsets = apriori(transactions, min\_support)

print("Частые наборы элементов:")

for itemset in frequent\_itemsets:

print(itemset)