|  |  |  |
| --- | --- | --- |
|  | Изображение выглядит как зарисовка, рисунок, символ, эмблема  Автоматически созданное описание |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| **Институт комплексной безопасности и специального приборостроения**  Кафедра КБ-14 «Технологии искусственного интеллекта и анализ данных»  Дисциплина «Введение в машинное обучение» | | |

**Отчёт**

«Практическая работа №6»

Выполнил:

Студент 3 курса

Группа БСБО-06-22

Шифр 22Б1240

Белявцева Екатерина Александровна

Москва, 2024

# Задание

Реализация алгоритма Apriori на языке Pyhton и в аналитической платформе deductor academic.

Реализуйте алгоритм поиска ассоциативных правил на Pyhton и выполните задания. В качестве альтернативы Pyhton можно использовать платформу deductor academic.

1. Выполните действия, описанные выше, используя различные параметры построения ассоциативных правил. Сравните полученные результаты, объясните их.
2. Ответьте на вопросы:

* какой товар с наибольшей достоверностью берут с вафлями?
* человек взял мед и сыры, какой один из товаров он скорее всего не возьмёт?
* назовите 5 самых популярных наборов товаров (в наборе может быть один или несколько товаров).
* опишите 4-5 ассоциативных правил, полученных в ходе выполнения работы.
* где еще, кроме торговли, можно использовать ассоциативные правила? Приведите примеры.

1. Используйте 5 различных комбинаций поддержки правила и достоверности (значимости) правил и исследуйте полученные результаты.
2. Составьте отчет из скриншотов последовательности Ваших действий.

# Ход работы

В ходе работы будет использоваться Python.

1. В начале будет реализован код для поиска ассоциативных правил. Будет использоваться алгоритм Apriori. Далее в листинге 1 будет представлен код для составления ассоциативных правил и их подсчёт при использовании различных параметров.

Листинг 1. Реализация алгоритма поиска ассоциативных правил

|  |
| --- |
| from collections import defaultdict from itertools import combinations  def load\_data(filename):  data = defaultdict(list)  with open(filename, "r", encoding="cp1251") as file:  next(file) # Пропускаем заголовок  for line in file:  check, product = line.strip().split("\t")  data[check].append(product)  return list(data.values())   def find\_frequent\_itemsets(transactions, min\_support):  item\_counts = defaultdict(int)  for transaction in transactions:  for item in transaction:  item\_counts[frozenset([item])] += 1  for size in range(2, len(transaction) + 1):  for combo in combinations(transaction, size):  item\_counts[frozenset(combo)] += 1  num\_transactions = len(transactions)  return {item: count for item, count in item\_counts.items() if count / num\_transactions >= min\_support}   def generate\_rules(frequent\_itemsets, min\_confidence, num\_transactions):  rules = []  for itemset in frequent\_itemsets:  if len(itemset) < 2:  continue  for item in itemset:  antecedent = frozenset([item])  consequent = itemset - antecedent  support = frequent\_itemsets[itemset] / num\_transactions  confidence = frequent\_itemsets[itemset] / frequent\_itemsets[antecedent]  if confidence >= min\_confidence:  rules.append((antecedent, consequent, support, confidence))  return rules   def display\_results(params, frequent\_itemsets, rules):  print(f"| {'Параметры':<20} | {'Частых наборов':<20} | {'Правил':<10} | {'Пример правила':<30} | {'Поддержка':<10} | {'Уверенность':<10} |")  print(f"{'-' \* 100}")  if rules:  example\_rule = rules[0]  print(f"| {params:<20} | {len(frequent\_itemsets):<20} | {len(rules):<10} | {set(example\_rule[0])} => {set(example\_rule[1]):<20} | {example\_rule[2]:<10.2f} | {example\_rule[3]:<10.2f} |")  else:  print(f"| {params:<20} | {len(frequent\_itemsets):<20} | {len(rules):<10} | {'Нет правил':<30} | {'-':<10} | {'-':<10} |")   filename = "Supermarket.txt" min\_support = 0.1 min\_confidence = 0.3  # Проведение экспериментов с параметрами parameters = [  {"min\_support": 0.1, "min\_confidence": 0.3},  {"min\_support": 0.1, "min\_confidence": 0.5},  {"min\_support": 0.2, "min\_confidence": 0.3},  {"min\_support": 0.2, "min\_confidence": 0.5}, ]  transactions = load\_data(filename) frequent\_itemsets = find\_frequent\_itemsets(transactions, min\_support) num\_transactions = len(transactions)  # Вывод результатов print(f"| {'min\_support':<12} | {'min\_confidence':<15} | {'Правил':<8} |") print(f"{'-' \* 43}") for params in parameters:  frequent\_itemsets = find\_frequent\_itemsets(transactions, params['min\_support'])  rules = generate\_rules(frequent\_itemsets, params['min\_confidence'], num\_transactions)  print(f"| {params['min\_support']:<12} | {params['min\_confidence']:<15} | {len(rules):<8} |") |

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1. Результат работы кода из листинга 1

В итоге чем меньше поддержка (min\_support) и уверенность (min\_confidence), тем больше появляется ассоциативных правил.

Из этого следует, что низкий порог поддержки позволяет выявлять редкие комбинации, но добавляет шум в результаты. Высокий порог уверенности сокращает количество правил, делая их более надёжными, но может исключить полезные закономерности.

2. Ответы на вопросы будут представлены ниже на рисунке 2. Для этого код будет модифицирован так, чтобы ответить сразу на все вопросы (кроме последнего).

Листинг 2. Ответы на вопросы

|  |
| --- |
| from collections import defaultdict from itertools import combinations import pandas as pd   def load\_data(filename):  data = defaultdict(list)  with open(filename, "r", encoding="cp1251") as file:  next(file)  for line in file:  check, product = line.strip().split("\t")  data[check].append(product)  return list(data.values())   def find\_frequent\_itemsets(transactions, min\_support):  item\_counts = defaultdict(int)  for transaction in transactions:  for item in transaction:  item\_counts[frozenset([item])] += 1  for size in range(2, len(transaction) + 1):  for combo in combinations(transaction, size):  item\_counts[frozenset(combo)] += 1  num\_transactions = len(transactions)  return {item: count for item, count in item\_counts.items() if count / num\_transactions >= min\_support}   def generate\_rules(frequent\_itemsets, min\_confidence, num\_transactions):  rules = []  for itemset in frequent\_itemsets:  if len(itemset) < 2:  continue  for item in itemset:  antecedent = frozenset([item])  consequent = itemset - antecedent  support = frequent\_itemsets[itemset] / num\_transactions  confidence = frequent\_itemsets[itemset] / frequent\_itemsets[antecedent]  if confidence >= min\_confidence:  rules.append((antecedent, consequent, support, confidence))  return rules   def find\_best\_association\_with\_wafer(rules):  max\_confidence = 0  best\_item = None  for antecedent, consequent, support, confidence in rules:  if 'ВАФЛИ' in antecedent:  for item in consequent:  if confidence > max\_confidence:  max\_confidence = confidence  best\_item = item  return best\_item, max\_confidence   def find\_rare\_items\_with\_honey\_and\_cheese(filename):  data = pd.read\_csv(filename, sep="\t", encoding="windows-1251", names=["Чек", "Товар"])   honey\_receipts = set(data[data["Товар"] == "МЕД"]["Чек"])  cheese\_receipts = set(data[data["Товар"] == "СЫРЫ"]["Чек"])  common\_receipts = honey\_receipts & cheese\_receipts   filtered\_data = data[data["Чек"].isin(common\_receipts) & ~data["Товар"].isin(["МЕД", "СЫРЫ"])]   item\_counts = filtered\_data["Товар"].value\_counts()   rare\_items = item\_counts[item\_counts == item\_counts.min()]  return rare\_items   def find\_top\_itemsets(frequent\_itemsets, num\_top=5):  sorted\_itemsets = sorted(frequent\_itemsets.items(), key=lambda x: x[1], reverse=True)  return sorted\_itemsets[:num\_top]   def display\_results(params, frequent\_itemsets, rules):  print(  f"| {'Параметры':<20} | {'Частых наборов':<20} | {'Правил':<10} | {'Пример правила':<30} | {'Поддержка':<10} | {'Уверенность':<10} |")  print(f"{'-' \* 100}")  if rules:  example\_rule = rules[0]  print(  f"| {params:<20} | {len(frequent\_itemsets):<20} | {len(rules):<10} | {set(example\_rule[0])} => {set(example\_rule[1]):<20} | {example\_rule[2]:<10.2f} | {example\_rule[3]:<10.2f} |")  else:  print(  f"| {params:<20} | {len(frequent\_itemsets):<20} | {len(rules):<10} | {'Нет правил':<30} | {'-':<10} | {'-':<10} |")  filename = "Supermarket.txt"  parameters = [  {"min\_support": 0.1, "min\_confidence": 0.3},  {"min\_support": 0.1, "min\_confidence": 0.5},  {"min\_support": 0.2, "min\_confidence": 0.3},  {"min\_support": 0.2, "min\_confidence": 0.5}, ] transactions = load\_data(filename) num\_transactions = len(transactions) # Вывод результатов print(f"| {'min\_support':<12} | {'min\_confidence':<15} | {'Правил':<8} |") print(f"{'-' \* 43}") for params in parameters:  frequent\_itemsets = find\_frequent\_itemsets(transactions, params['min\_support'])  rules = generate\_rules(frequent\_itemsets, params['min\_confidence'], num\_transactions)  print(f"| {params['min\_support']:<12} | {params['min\_confidence']:<15} | {len(rules):<8} |")   best\_item, max\_confidence = find\_best\_association\_with\_wafer(rules) if best\_item:  print(f"\nТовар с наибольшей уверенностью, связанный с вафлями: {best\_item} с уверенностью {max\_confidence:.2f}") else:  print("Нет товара с наибольшей уверенностью, связанного с вафлями.")   rare\_items = find\_rare\_items\_with\_honey\_and\_cheese(filename) if not rare\_items.empty:  print("\nТовары, которые реже всего встречаются с медом и сырами:")  print(rare\_items.reset\_index().to\_string(index=False, header=True)) else:  print("\nНет товаров, которые реже всего встречаются с медом и сырами.")   frequent\_itemsets = find\_frequent\_itemsets(transactions, min\_support=0.1) top\_itemsets = find\_top\_itemsets(frequent\_itemsets, num\_top=5) print("\nТоп-5 самых популярных наборов товаров:") for itemset, count in top\_itemsets:  print(f"{set(itemset)}: встречается {count} раз")   print("\nПример ассоциативных правил:") for i, (antecedent, consequent, support, confidence) in enumerate(rules[:5]):  print(f"{i+1}. {set(antecedent)} => {set(consequent)} | Поддержка: {support:.2f} | Уверенность: {confidence:.2f}") |

Изображение выглядит как текст, снимок экрана

Автоматически созданное описаниеРисунок 2. Результат работы кода из листинга 2, то есть ответы на вопросы

Кроме торговли ассоциативные правила можно использовать, например в медицине. Например, выявление факторов риска заболеваний: «Если у пациента есть кашель и высокая температура, вероятнее всего, это грипп», или подбор оптимального лечения: «Если пациент имеет гипертонию и диабет, то назначение гипотензивных препаратов с низким уровнем побочных эффектов предпочтительнее».

Также ассоциативные правила можно использовать в финансовой аналитике. Например, предсказание финансовых рисков: «Если клиент часто переводит деньги в определённые регионы, это может быть признаком мошеннической деятельности», или предсказание стоимости активов: «Если в прошлые периоды наблюдался рост цены нефти и снижение цен на золото, в следующем периоде золото также может подешеветь»

3. Чтобы исследовать результаты с различными комбинациями поддержки и достоверности, можно изменить параметры min\_support и min\_confidence в коде. Остальной код остался неизменным

Листинг 3. Изменения в коде для вывода результатов для задания 3

|  |
| --- |
| filename = "Supermarket.txt"  parameters = [  {"min\_support": 0.05, "min\_confidence": 0.3},  {"min\_support": 0.1, "min\_confidence": 0.5},  {"min\_support": 0.2, "min\_confidence": 0.3},  {"min\_support": 0.3, "min\_confidence": 0.7},  {"min\_support": 0.15, "min\_confidence": 0.6}, ]  transactions = load\_data(filename) num\_transactions = len(transactions)  # Вывод результатов print(f"| {'min\_support':<12} | {'min\_confidence':<15} | {'Правил':<8} |") print(f"{'-' \* 43}") for params in parameters:  frequent\_itemsets = find\_frequent\_itemsets(transactions, params['min\_support'])  rules = generate\_rules(frequent\_itemsets, params['min\_confidence'], num\_transactions)  print(f"| {params['min\_support']:<12} | {params['min\_confidence']:<15} | {len(rules):<8} |")  if rules:  for rule in rules[:5]:  print(f"Правило: {set(rule[0])} => {set(rule[1])}, поддержка: {rule[2]:.2f}, уверенность: {rule[3]:.2f}")  else:  print("Нет правил для данной комбинации параметров.") |

В результате низкой поддержки и низкой уверенности этого сочетания будет найдено много редких правил, которые могут быть статистически значимыми, но не столь сильными.

В случае высокой поддержки и высокой уверенности будет найдено меньше правил, но они будут более надежными и значимыми.

Комбинация средней поддержки и уверенности обеспечит баланс между количеством и качеством правил, что может быть полезным для поиска интересных закономерностей, которые не слишком редки, но и не слишком популярны.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3. Результат работы кода из листинга 2 с учётом изменения вывода (листинга 3)

«КЕТЧУПЫ, СОУСЫ, АДЖИКА» ⇔ «МАКАРОННЫЕ ИЗДЕЛИЯ» –доминирующая ассоциация во всех комбинациях, что подчеркивает их взаимосвязь в покупках.

«ЧАЙ» появляется в нескольких связях, но его уверенность относительно других товаров низка на начальных этапах (min\_confidence 0.3), но растет с увеличением достоверности.

На более строгих настройках, таких как min\_support = 0.3 и min\_confidence = 0.7, появляется сильная связь между «МЕД» и «ЧАЙ» с высокой уверенностью.

Популярные товары, такие как «ВАФЛИ» и «СЫРЫ», также начинают показывать высокую уверенность в их ассоциациях с «ЧАЕМ» на более высоких значениях достоверности, что может свидетельствовать о высоком интересе к этим продуктам в контексте покупок.