ДОДАТОК А

Графічні матеріал

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

«ЗАТВЕРДЖУЮ»

Керівник атестаційної роботи

_____проф. Ситніков Д.Е.

(підпис)

РОЗРОБКА МЕТОДІВ CRM НА БАЗІ ГЕНЕРАЦІЇ АСОЦІАТИВНИХ ПРАВИЛ ТА ЇХ ВИКОРИСТАННЯ В ІНФОРМАЦІЙНІЙ СИСТЕМІ ПРОДАЖУ КВІТІВ -

Графічний матеріал

ЛИСТЗАТВЕРДЖЕННЯ

ГЮИК.506120.012 34 01- 01 12 01 – ЛУ

РОЗРОБИЛА

ст. гр. ІТПм-19-1

Шубіна Н.А.

2020 р.

ЗАТВЕРДЖЕНО

ГЮИК.506120.012 34 01 - 01 12 01 – ЛУ

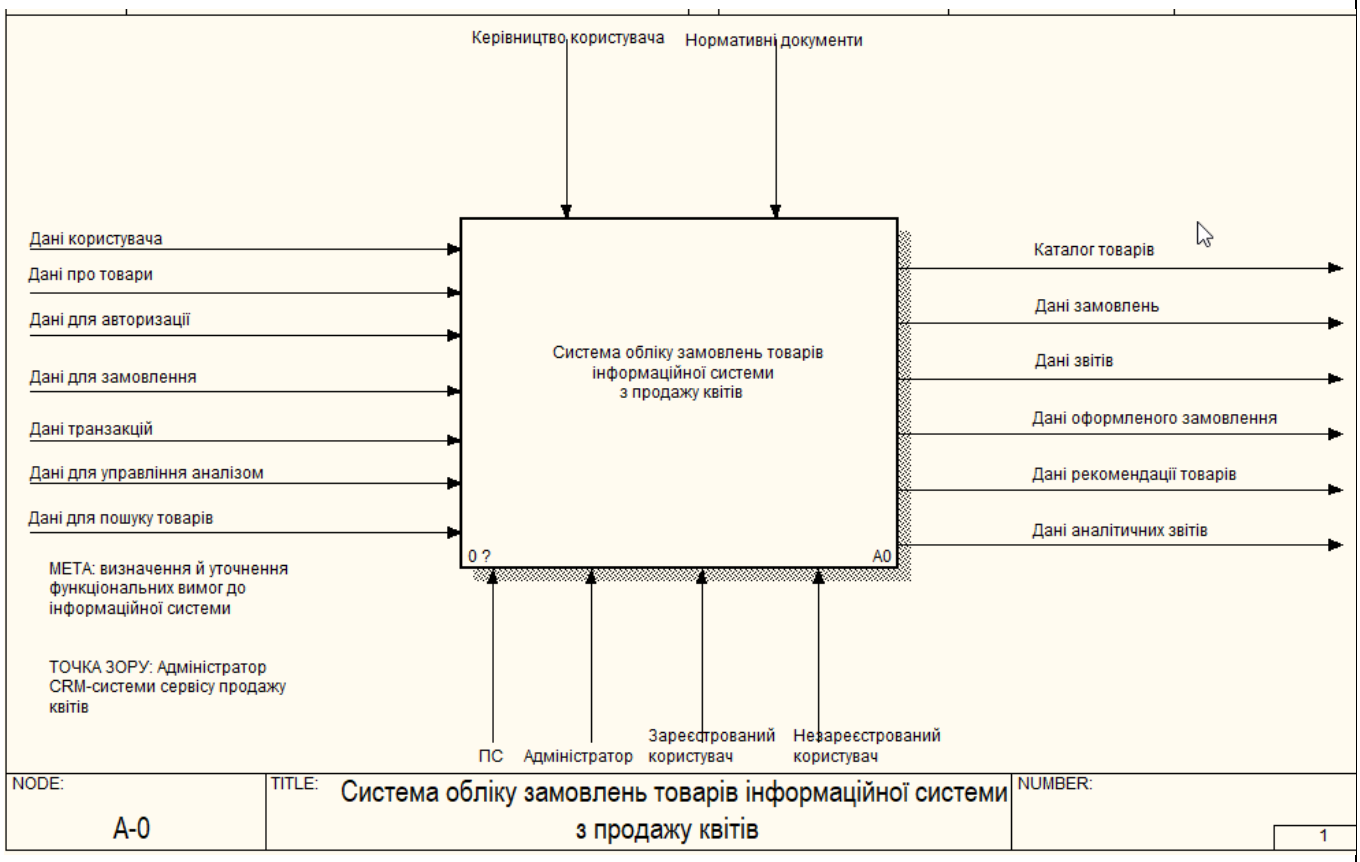
РОЗРОБКА МЕТОДІВ CRM НА БАЗІ ГЕНЕРАЦІЇ АСОЦІАТИВНИХ ПРАВИЛ ТА ЇХ ВИКОРИСТАННЯ В ІНФОРМАЦІЙНІЙ СИСТЕМІ ПРОДАЖУ КВІТІВ

Графічний матеріал

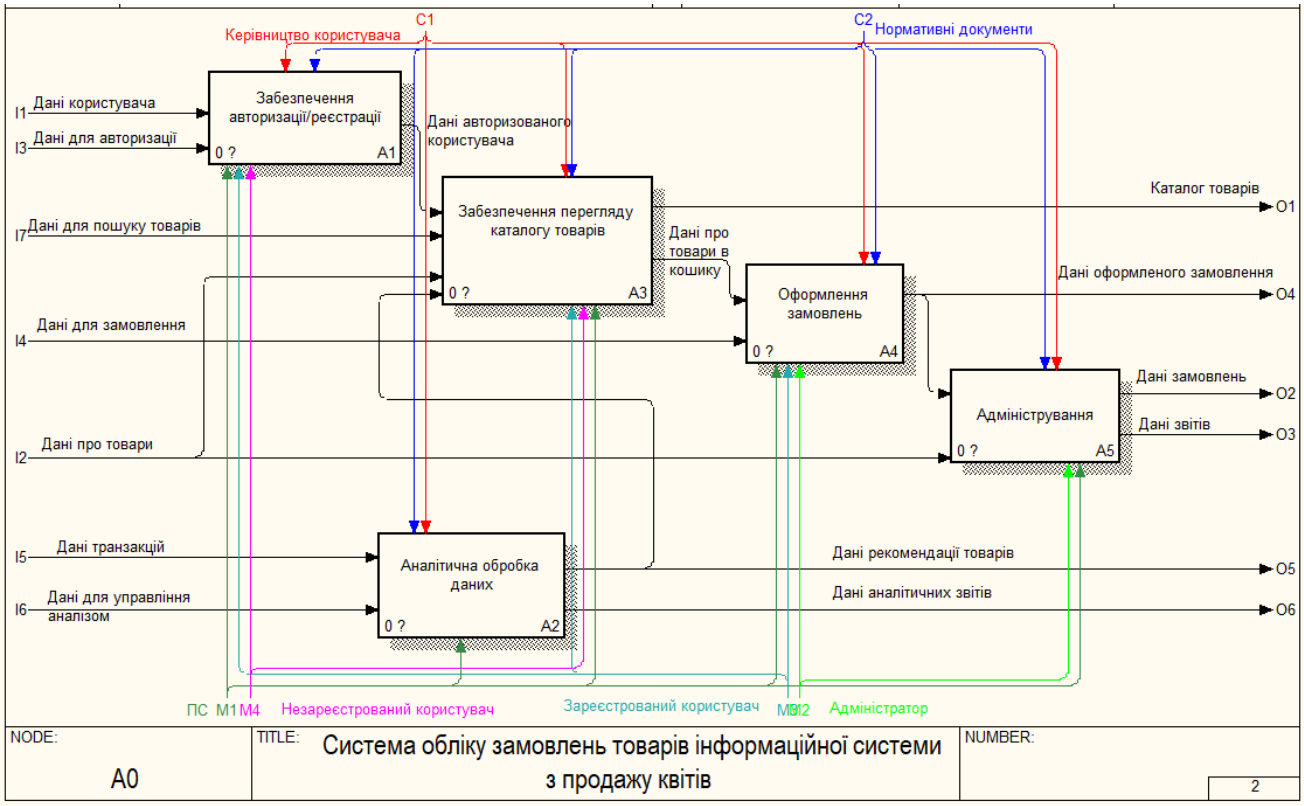
ГЮИК.506120.012 34 01 – 01 12 01 ЛУ

Листів 5

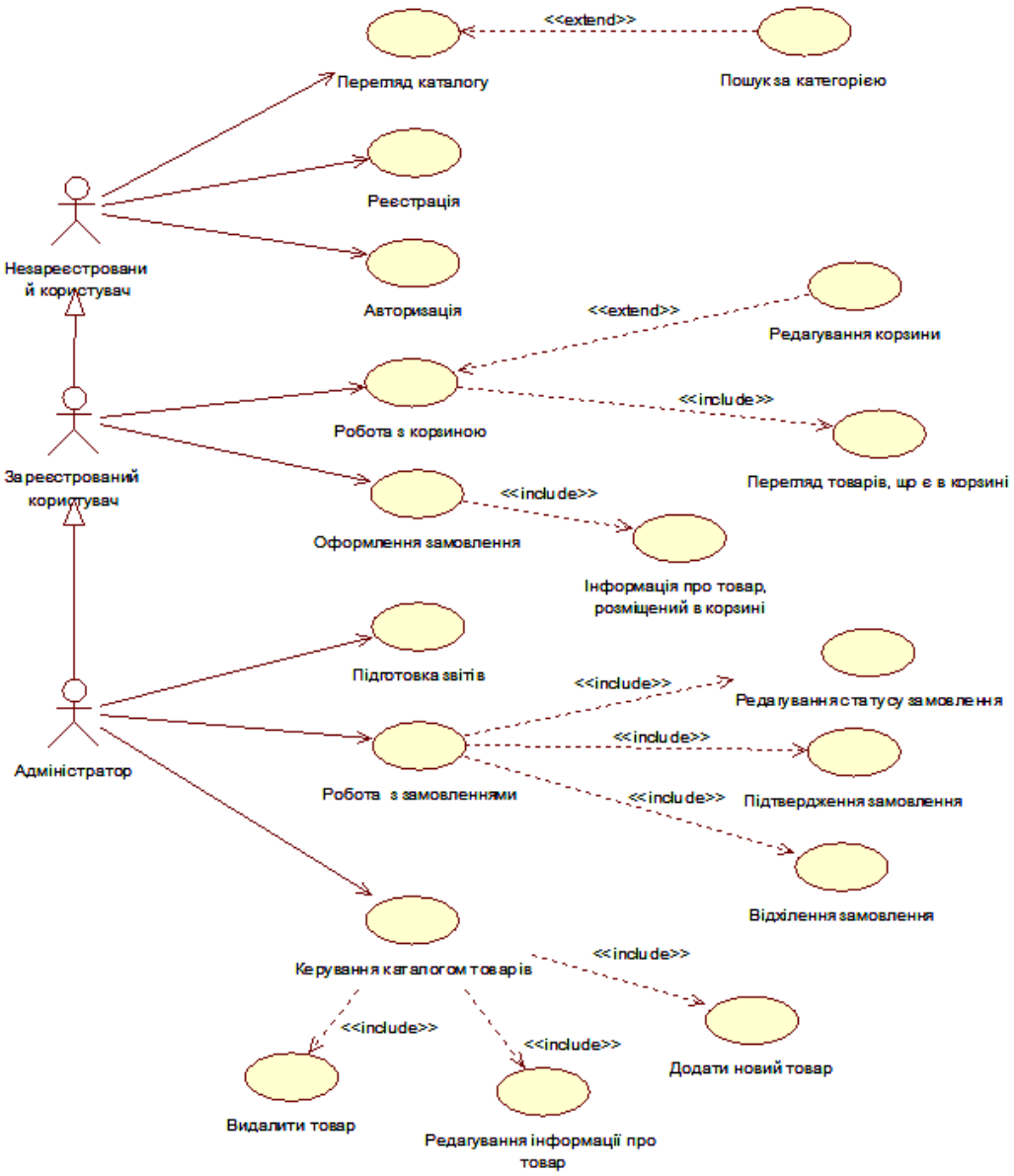
2020



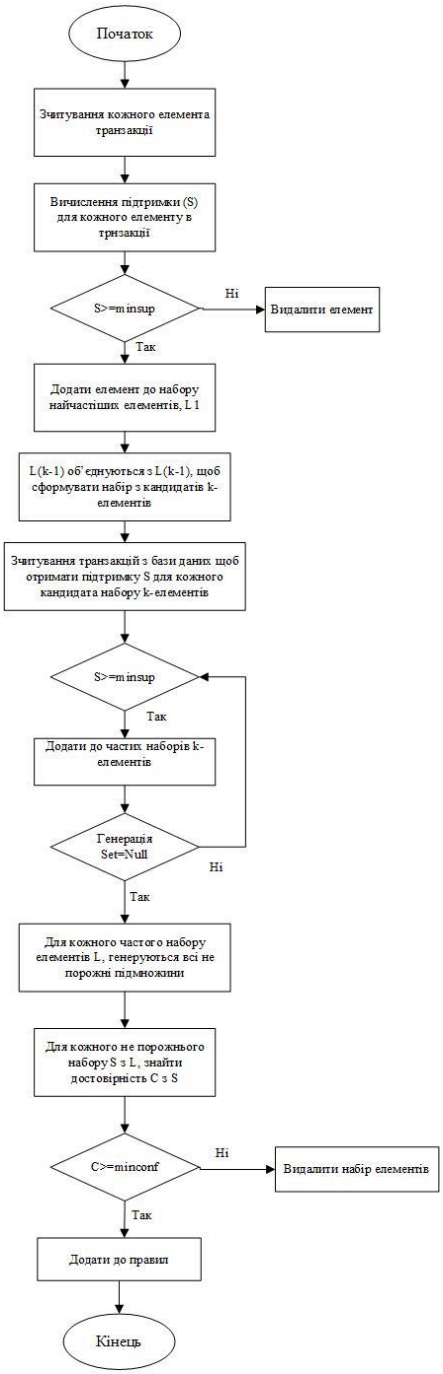
Розробила	Шубіна Н.А.			Концептуальна ддіаграма	
Перевірив	Ситніков Д.Е.				
Н.Контр.	Ситніков Д.Е.				
				ІТПм-19-1	Лист 1
Затвердив	Гребеннік І. В.			СТ	Листів



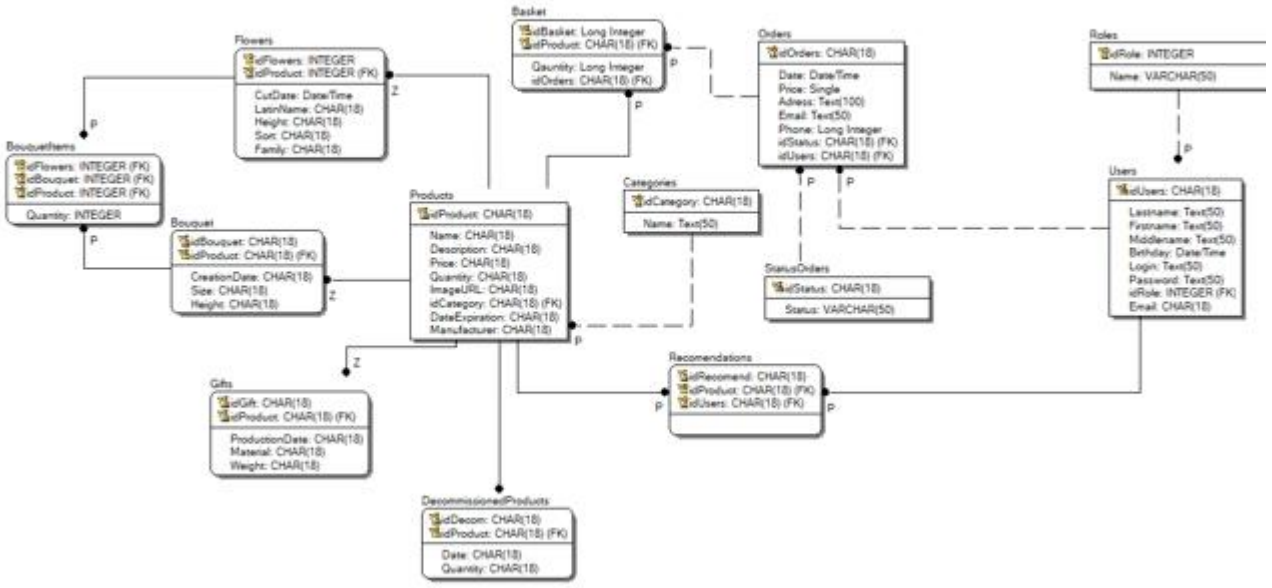
Розробила	Шубіна Н.А.			Декомпозиція головної функції системи	
Перевірив	Ситніков Д.Е.				
Н.Контр.	Ситніков Д.Е.				
				ІТПм-19-1	Лист 2
Затвердив	Гребеннік І. В.			СТ	Листів



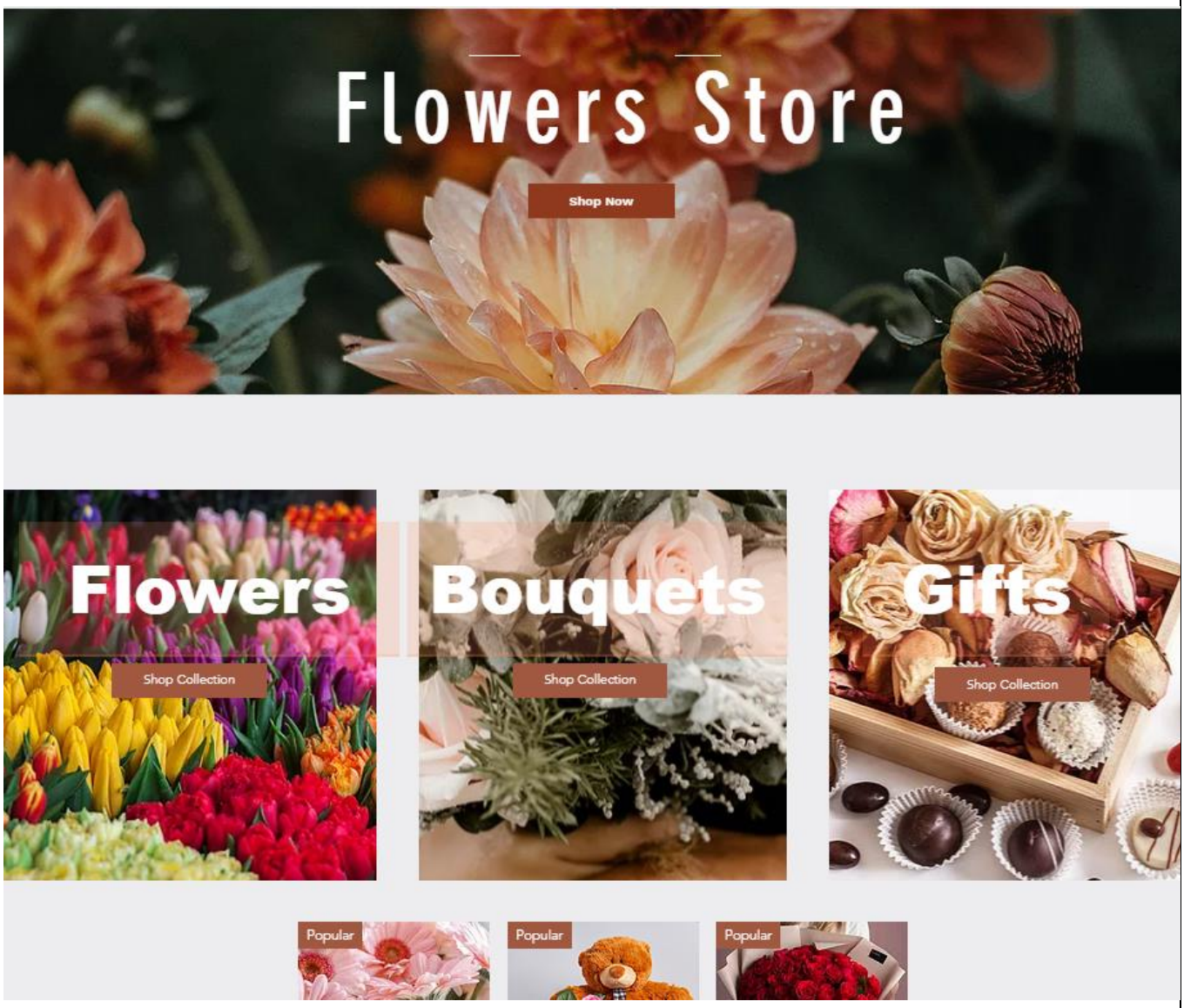
Розробила	Шубіна Н.А.			Use case діаграма	
Перевірив	Ситніков Д.Е.				
Н.Контр.	Ситніков Д.Е.				
				ІТПм-19-1	Лист 3
Затвердив	Гребеннік І. В.			СТ	Листів



Розробила	Шубіна Н.А.			Узагальнений алгоритм Apriori	
Перевірив	Ситніков Д.Е.				
Н.Контр.	Ситніков Д.Е.				
				ІТПм-19-1	Лист 4
Затвердив	Гребеннік І. В.			СТ	Листів



Розробила	Шубіна Н.А.			ER діаграма бази даних	
Перевірив	Ситніков Д.Е.				
Н.Контр.	Ситніков Д.Е.				
				ІТПм-19-1	Лист 5
Затвердив	Гребеннік І. В.			СТ	Листів



Розробила	Шубіна Н.А.			Головна сторінка з реалізованим блоком рекомендацій	
Перевірив	Ситніков Д.Е.				
Н.Контр.	Ситніков Д.Е.				
				ІТПм-19-1	Лист 6
Затвердив	Гребеннік І. В.			СТ	Листів

ДОДАТОК Б
Текст програми

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

«ЗАТВЕРДЖУЮ»

Керівник атестаційної роботи

_____проф. Ситніков Д.Е.

(підпис)

РОЗРОБКА МЕТОДІВ CRM НА БАЗІ ГЕНЕРАЦІЇ АСОЦІАТИВНИХ ПРАВИЛ ТА ЇХ ВИКОРИСТАННЯ В ІНФОРМАЦІЙНІЙ СИСТЕМІ ПРОДАЖУ КВІТІВ -

Текст програми

ЛИСТЗАТВЕРДЖЕННЯ

ГЮИК.506120.012 34 01- 01 12 01 – ЛЗ

РОЗРОБИЛА

ст. гр. ІТПм-19-1

Шубіна Н.А.

2020 р.

ЗАТВЕРДЖЕНО

ГЮИК.506120.021 34 01 - 01 12 01 – ЛЗ

РОЗРОБКА МЕТОДІВ CRM НА БАЗІ ГЕНЕРАЦІЇ АСОЦІАТИВНИХ ПРАВИЛ ТА ЇХ ВИКОРИСТАННЯ В ІНФОРМАЦІЙНІЙ СИСТЕМІ ПРОДАЖУ КВІТІВ

Текст програми

ГЮИК.506120.012 34 01 – 01 12 01

Листів 5

2020

Реалізація асоціативного алгоритму Apriori з урахуванням строків зберігання та сумісності товарів:

```
from scipy.stats import hmean
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer
import argparse
import itertools
import json
import numpy
import pandas

class Apriori:

	def __init__(self, min_support: float = 0., max_support: float = 1.):
		self.__min_support = min_support
		self.__max_support = max_support
		self.__dataSet = list()
		self.__itemsTitles = list()
	self.recommendations = list()

	def __support(self, a):
		count = 0
		for transaction in self.__dataSet:
			if all(item in transaction for item in a):
				count += 1
		return float(count) / float(len(self.__dataSet))

	def __confidence(self, a, b):
		if self.__support(a) == 0:
			return 0
		return self.__support(self.__union(a, b)) / self.__support(a)

	def __lift(self, a, b):
		if self.__support([b]) == 0:
			return 0
```

```
return self.__confidence(a, b) / self.__support([b])

	def __conviction(self, a, b):
		if self.__confidence(a, b) == 1:
			return 0
		return (1 - self.__support([b])) / (1 - self.__confidence(a, b))

	def __find_frequencies(self):
		frequent_itemsets = []
		N = len(self.__itemsTitles)
		candidates = self.__itemsTitles
		for k in range(1, N + 1):
			combinations = list(itertools.combinations(candidates, k))
			frequent_combs = []
		for comb in combinations:
			output = "comb: " + str(comb) + " sup: " +
			str(self.__support(comb))
			print(output)
			if self.__min_support <= self.__support(comb) <=
				self.__max_support:
				frequent_combs.append(comb)
				candidates = []
			for transaction in frequent_combs:
				frequent_itemsets.append(transaction)
			for item in transaction:
			if item not in candidates:
				candidates.append(item)
				print('frequent itemsets:', frequent_itemsets)
			return frequent_itemsets
def fit(self, dataSet):
 # составление списка часто встречающихся сочетаний товаров
		self.__dataSet = dataSet
		self.__itemsTitles = MultiLabelBinarizer().fit(dataSet).classes_
		frequent_itemsets = self.__find_frequencies()
 # поиск ассоциативных правил
		N = len(self.__itemsTitles)
		for k in range(len(frequent_itemsets)):
			itemset_A = frequent_itemsets[k]
```

```
            for i in range(N):
                  item_B = self.__itemsTitles[i]
            if item_B in itemset_A:
            continue
                  print("rec: " + str(itemset_A) + "->" + str(item_B),
                  "conf: " + str(self.__confidence(itemset_A, item_B)),
                  "lift: " + str(self.__lift(itemset_A, item_B)),
                  "conv: " + str(self.__conviction(itemset_A, item_B)))
            if self.__lift(itemset_A, item_B) > 1:
                  self.recommendations.append(dict(itemset=itemset_A,
                  item=item_B,

                  conf=self.__confidence(itemset_A, item_B),
                  lift=self.__lift(itemset_A,
                  item_B),
                  conv=self.__conviction(itemset_A, item_B)))
            return self.recommendations


      def resultict(self, dataSet, n_recs: int = 1, metric: str='confidence'):
            try:
            if metric == 'confidence':
                  self.recommendations.sort(key=lambda x:
                  self.__confidence(x['itemset'], x['item']), reverse=True)
                  elif metric == 'lift':
                        self.recommendations.sort(key=lambda x: self.__lift(x['itemset'],
                  x['item']), reverse=True)
                  elif metric == 'conviction':
                        self.recommendations.sort(key=lambda x:
                        self.__conviction(x['itemset'], x['item']), reverse=True)
            elif metric == 'hmean':
                  self.recommendations.sort(key=lambda x:
                  hmean([self.__confidence(x['itemset'], x['item']),
                  self.__lift(x['itemset'], x['item']),
                  self.__conviction(x['itemset'], x['item'])]), reverse=True)
            else:
                  raise Exception(metric)
                  except Exception as inst:
```

```
                    print("ERROR. Unknown value for atribute 'metric': " + inst.args[0])
                    exit(1)
                    result = list()
                    for itemset in dataSet:
                          result = list()
                          for rec in self.recommendations:
                          if all(rec_item in itemset for rec_item in rec['itemset']) \
                          and rec['item'] not in itemset \
                          and rec['item'] not in result:
                                result.append(rec['item'])
                                if len(result) >= n_recs:
                                      break
                          result.append(result)
                          return result


      @staticmethod
      def __union(a, b):
            if type(b) == list:
                  return list(set().union(a, b))
                  return list(set().union(a, [b]))


      def backtest(self, X, y):
            return numpy.array([self.__conviction(a, b) for a, b in zip(X, y)]).mean()
            if __name__ == '__main__':
                  parser    =    argparse.ArgumentParser(description='Program    provides
recommendations',
                  usage='python')
                  parser.add_argument('-minSupport', type=float, default=0., help='Minimum
support')
                  parser.     add_argument('-maxSupport',     type=float,     default=1.,
help='Maximum support')
                  parser.add_argument('filepath', help='file with transactions')
                  parser.add_argument('-nRecom',  type=int,  default=1,  help='Number  of
recommendations for one transaction')
                  parser.add_argument('-metric',  default='confidence',  help='Value  of
metric for recommendation search')
```

```
args = parser.parse_args()

with open(args.filepath, encoding='utf-8') as f:
      data = json.load(f)
dataSet = []
for i in range(len(data)):
transaction = [data[i]['items'][j]['commodityName'] for j in
range(len(data[i]['items']))]
            dataSet.append(transaction)

train, test = train_test_split(dataSet, test_size=0.005)
ap = Apriori(args.minSupport, args.maxSupport)
ap.fit(train)
print(ap.recommendations)
result = ap.resultict(test, n_recs=args.nRecom, metric=args.metric)
df = pandas.DataFrame(numpy.array(result))
df.to_csv("result.csv")
```

ДОДАТОК В
Відомість атестаційної роботи

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК.506120.012 ПЗ	Пояснювальна записка	64 аркушів
2	ГЮИК.506120.012 34 01 ИЗ	Текст програми	5 аркушів
3	ГЮИК.506120.012 – 01 12 01		
		Графічні документи	
4		Концептуальна діаграма	1 аркуш
		системи	
5		Діаграма декомпозиції	1 аркуш
		головної функції системи	
6		Діаграма варіантів	1 аркуш
		використання системи	
7		Узагальнений алгоритм генерації	1 аркуш
		асоціативних правил Apriori	
8		ER діаграма бази даних	1 аркуш
		інформаційної системи	

					ГЮИК.507200.012 ДЗ			
Змін.	*Лист*	*№ докум.*	*Підпис*	*Дата*				
Розробив		*Шубіна Н.А..*			*Розробка методів CRM на базі генерації асоціативних правил та їх використання в інформаційній системі продажу квітів*		Лист	Листів
Перевірив		*Ситніков Д.Е.*					*1*	*1*
Н. Контр.						*ХНУРЕ Кафедра СТ*		
Затвердив		*Гребеннік І.В.*						