Apriori Algorithm 결과 보고서

이름 : 박병규

학번 : 20181612

Association rule의 정의에 따라서 I, j에 따라서 I와 j가 있는 장바구니의 수를 count를 해주고 confidence를 구해주는 함수를 정의해주었다.

Support 함수에서는 data가 엄청 큰 상황임을 가정하고 파일을 새로 읽어가며 support값을 구했다.

```
def support(I, j) :
    cnt1 = 0 # support union I | j
    cnt2 = 0 # support I
    with open("groceries.csv", "r") as f :
        for line in f :
            basket = line.strip().split(",")
            basket = frozenset(basket)
            #print(basket)
            if len(I | j | basket) == len(basket) :
                cnt1 += 1
                cnt2 += 1
            elif len(I|basket) == len(basket) :
                cnt2 += 1
    return cnt1, cnt2

def conf(I, j) :
    cnt1, cnt2 = support(I, j)
    ret = cnt1/cnt2
    return ret
```

S = 100, c는 0.5 값을 설정하고 과제를 진행하였다.

실습을 통해 구한 freq_itemsets_all집합에서 I와 j를 뽑아내고 위에 구현한 두개의 함수를 이용하여 confidence를 구하였다. 그리고 confidence가 0.5가 넘는 경우에 대해 print를 진행하였다.

test진행을 위해 test_list에 I, j, confidence값을 넣어주었다.

```
test_list = []
for target in freq_itemsets_all :
    cnt1 = 0
    cnt2 = 0
    for i in range(1,len(target)) :
        for j in combinations(target, i) :
            j = frozenset(j)
            I = target - j
            #print(I)
            c = conf(I, j)
            # c값 설정
            if c >= 0.5 :
                test_list.append([I,j,round(c,2)])
                print(I, j, round(c,2))
apriori_end = time.time()

print(f"apriori run time: {apriori_end - apriori_start:.6f} sec")
```

Apriori 알고리즘을 통해 나온 test_list값이 맞는지 틀린지 계산해보았다.

Test 함수는 이번 과제의 검토용으로 만든거기 때문에 메모리는 고려를 하지 않았다.

아래 함수는 groceries.csv 파일을 dataframe으로 읽고 앞서 구한 confidence값이 맞는지 파일을 쭉 읽어가면서 계산을 했다.

계산 결과가 맞으면 correct 틀리면 fail을 출력하도록 하였다.

```
def test(test_list) :
    print()
    print()
    print("-------------------------test start-------------------------")
    file_path = 'C:/Users/pbk54/Desktop/3학년1학기/빅최기/과제3/groceries.csv'

    with open(file_path, encoding="utf8") as f :
        lines = f.readlines()
    #print(len(lines))

    len_list = []
    for _ in range(len(lines)) :
        len_list.append(len(lines[_].split(',')))

    Counter(len_list)

    f = open(file_path, encoding='utf8')
    reader = csv.reader(f)
    csv_list = []
    for l in reader :
        csv_list.append(l)
    f.close()
    df = pd.DataFrame(csv_list)
    test_start = time.time()
    for j in range(len(test_list)) :
        a = set(test_list[j][0])
        b = set(test_list[j][1])
        c = test_list[j][2]
        cnt1 = 0
        cnt2 = 0
        for i in range(len(df)) :
            if len(a | b | set(df.iloc[i])) == len(set(df.iloc[i])) :
                cnt1 += 1
                cnt2 += 1
            elif len(a| set(df.iloc[i])) == len(set(df.iloc[i])) :
                cnt1 += 1
        if c == round(cnt2/cnt1,2) :
            print(test_list[j], "correct")
        else :
            print("fail")
    test_end = time.time()
    print(f"test run time: {test_end - test_start:.6f} sec")
```

실습 결과

Apriori 알고리즘으로 구한 값과 test로 구한 값들이 다 일치함을 알 수 있었다.

Confidence는 소수 둘째자리까지 구했다.

그리고 test는 파일을 dataframe으로 읽는 과정에서 시간이 더 걸릴 수도 있었지만 test_list에 대해 즉 14번만을 반복을 했는데 apriori의 실행시간보다 2배이상 길게 나타났다.