Belirli ögeler arasındaki ilişkileri öğrenmek için kullanılan kurallardır. Genel amacı olarak iki ögenin birlikte görülme olasılığını inceleyen kurallardır.

**İlişkilendirme kuralları**, özellikle **pazar sepeti analizi** gibi uygulamalarda çok yaygındır. Pazar sepeti analizi ile bir marketin kasasında toplanan verilerde, hangi ürünlerin sıklıkla birlikte alındığını incelemek istiyoruz. Örneğin: “Ekmek” ve “Süt” ürünlerinin birlikte sıklıkla alındığını fark edebiliriz. İlişkilendirme kuralları bu tür ilişkileri keşfeder ve şu şekilde ifade edilir:

Kural: “Eğer Ekmek alırsanız, büyük olasılıkla Süt de alırsınız.”

Bunu matematiksel olarak şöyle ifade edebiliriz:

* A (Ekmek) → B (Süt)

Bu, Ekmek alındığında, Süt’ün de alınma olasılığının yüksek olduğunu belirten bir ilişkilendirme kuralıdır.

**Destek Oranı:**Bir öge kümesinin veri kümesindeki sıklığını ifade eder. Yani belirli bir ürünün veya ürün grubunun ne sıklıkla birlikte satın alındığını gösterir.

**Güven Oranı:**Bir kuralın ne kadar doğru olduğunu ifade eder.

**Algoritmanın Uygulama Aşamaları:**

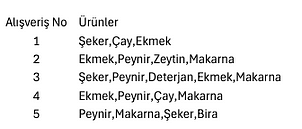
1- Destek Oranı X Veri Sayısı ile “Destek Sayısını” bulabiliriz.

2- Ürünlere göre sayı tablosu oluşturmalıyız. -Destek sayısına göre veriyi çıkartmalı-

3- Çıkan ürünleri sırasıyla ikili (oluşan durumlara göre bu sayılar arttırmalıyız) kombinasyon ile grupla.

4- En son oluşan gruba göre kuralları oluşturmalıyız. — Güven oranına göre kontrol etmeliyiz. —

**Örnek Uygulama**



[1] Örnek Verimiz

Destek(Eşik) = % 60

Güven(Eşik) = % 75

# Verilen veride veri sayısı 5 ve yine verilen destek oranı %60 olduğundan eşik sayısı 0.6 x 5 = 3 olarak hesaplanır.



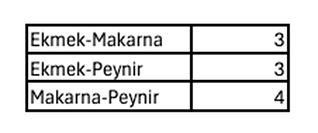
Eşik Sayısına göre de üçün altında olanları bir sonraki aşamaya göndermeden eliyoruz.



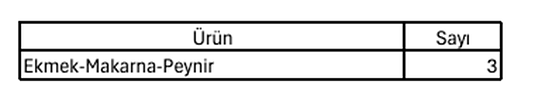
2. Aşamada bu seçilen ürünleri ikili grupluyoruz.



Bu tabloya göre tekrardan eşik değerine göre tekrardan eleme yapmamız gerekiyor.



Elimizde oluşan bu ürünleri üçlü gruplamamız gerekiyor.



Sayı(A,B) = Sayı(Ekmek,Makarna,Peynir) = 3

Destek Oranı kontrolü için ‘Sayı(A,B) / N’ bu formülü ve verilen destek oranı ile eşit veya yüksek olmalı.

3/5 = 0.60 = %60 verilen destek oranı ile eşit olduğu için kural yazma durumuna geçebiliriz.

**Güven Oranına Göre Yorumlar**

{Ekmek,Makarna,Peynir}

Sayı(A,B) / Sayı(A) ile verilen güven oranı kıyaslanıp yorumlayabilmemiz gerekir.

*Güven(Ekmek,Makarna => Peynir) | ile| 3/3 = 1 = %100*

~ “Eğer bir müşteri Ekmek ve Makarna alıyorsa, ona kesin olarak Peynir de satılacaktır. Bu, çok güçlü bir ilişkiyi gösterir.”

*Güven(Ekmek => Peynir,Makarna) | 3/4 = 0.75 = %75*

~ “Ekmek satın alırken, genellikle Peynir ve Makarna da alınıyor ama bu durum her zaman geçerli değil. Yani, Ekmek alan birinin bu iki ürünü de alması %75 olasılıkla gerçekleşiyor.”

*Güven(Peynir => Ekmek,Makarna) |3/4 = 0.75 = %75*

~ Peynir almanın, Ekmek ve Makarna alma olasılığı üzerinde belirli bir etkisi var, ancak her zaman bu ikisini birlikte almayacaklardır. Ancak %75 gibi yüksek bir güven oranı, güçlü bir ilişkiden söz edilebilir.

**Kod İle Apriori Algoritması**

*Kütüphaneler*

import csv # CSV çıktısı verebilmek için   
from itertools import combinations # Belirli boyuttta eleman kombinasyonları oluşturmak için

*Destek Değerini Hesaplayan Fonksiyon*

def calculate\_support(data, itemset):  
 count = 0  
 for transaction in data:  
 if itemset.issubset(transaction):  
 count += 1  
 return count / len(data)

*Adayları Oluşturan Fonksiyon*

def generate\_candidates(frequent\_itemsets, k):  
 candidates = set() # Boş aday kümesi  
 frequent\_items = list(frequent\_itemsets)  
 for i in range(len(frequent\_items)):  
 for j in range(i + 1, len(frequent\_items)):  
 union\_set = frequent\_items[i] | frequent\_items[j]  
 if len(union\_set) == k:  
 candidates.add(frozenset(union\_set))  
 return candidates

*Genel Algoritmayı Sağlayan Fonksiyon*

def apriori(data, min\_support):  
 transactions = [set(transaction) for transaction in data]  
 items = {frozenset([item]) for transaction in transactions for item in transaction}  
 frequent\_itemsets = {item for item in items if calculate\_support(transactions, item) >= min\_support}  
   
 k = 2  
 result = list(frequent\_itemsets)  
 while frequent\_itemsets:  
 candidates = generate\_candidates(frequent\_itemsets, k)  
 frequent\_itemsets = {itemset for itemset in candidates if calculate\_support(transactions, itemset) >= min\_support}  
 result.extend(frequent\_itemsets)  
 k += 1  
 return result

*Kuralları Yazdıran Fonksiyon*

def generate\_rules(frequent\_itemsets, data, min\_confidence):  
 transactions = [set(transaction) for transaction in data]  
 rules = []  
 for itemset in frequent\_itemsets:  
 subsets = [frozenset(x) for x in combinations(itemset, len(itemset) - 1)]  
 for subset in subsets:  
 remain = itemset - subset  
 if remain:  
 confidence = calculate\_support(transactions, itemset) / calculate\_support(transactions, subset)  
 if confidence >= min\_confidence:  
 rules.append((subset, remain, confidence))  
 return rules

*CSV’den Okuyan Fonksiyon*

def read\_transactions\_from\_csv(file\_path):  
 with open(file\_path, 'r') as file:  
 reader = csv.reader(file)  
 return [row for row in reader]

*CSV’ye Yazdıran Fonksiyon*

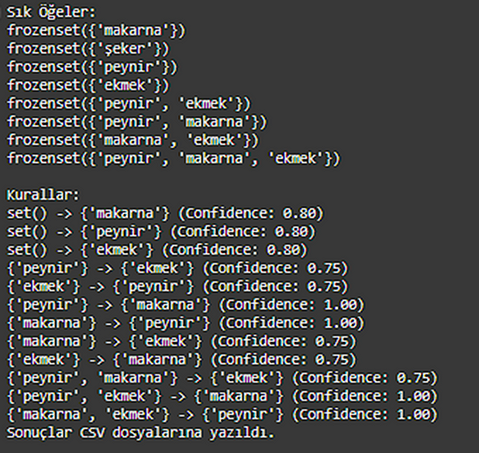
def write\_results\_to\_csv(frequent\_itemsets, rules, itemsets\_file, rules\_file):  
 with open(itemsets\_file, 'w', newline='') as file:  
 writer = csv.writer(file)  
 writer.writerow(["Frequent Itemsets"])  
 for itemset in frequent\_itemsets:  
 writer.writerow([", ".join(itemset)])  
  
 with open(rules\_file, 'w', newline='') as file:  
 writer = csv.writer(file)  
 writer.writerow(["Antecedent", "Consequent", "Confidence"])  
 for rule in rules:  
 writer.writerow([", ".join(rule[0]), ", ".join(rule[1]), f"{rule[2]:.2f}"])

*Örnek Kullanım*

input\_csv = '/content/Transaction.csv'  
output\_itemsets\_csv = 'frequent\_itemsets.csv'  
output\_rules\_csv = 'association\_rules.csv'  
  
data = read\_transactions\_from\_csv(input\_csv)  
  
min\_support = 0.5  
min\_confidence = 0.7  
  
frequent\_itemsets = apriori(data, min\_support)  
  
print("Sık Öğeler:")  
for itemset in frequent\_itemsets:  
 print(itemset)  
  
rules = generate\_rules(frequent\_itemsets, data, min\_confidence)  
print("\nKurallar:")  
for rule in rules:  
 print(f"{set(rule[0])} -> {set(rule[1])} (Confidence: {rule[2]:.2f})")  
  
write\_results\_to\_csv(frequent\_itemsets, rules, output\_itemsets\_csv, output\_rules\_csv)  
  
print("Sonuçlar CSV dosyalarına yazıldı.")

*Öneri sistemi*

def generate\_recommendations(rules, user\_input):  
 recommendations = []  
 for antecedent, consequent, confidence in rules:  
 if user\_input.issubset(antecedent): # Kullanıcının girdisi antecedent'te varsa  
 recommendations.append((consequent, confidence))  
 recommendations.sort(key=lambda x: x[1], reverse=True) # Güvene göre sırala  
 return recommendations  
  
user\_input = input("Lütfen öneri için bir öğe giriniz (örnek: 'milk'): ").split(',')  
user\_set = set(user\_input)  
recommendations = generate\_recommendations(rules, user\_set)  
print(recommendations)



[2] Çıktılarımız