# CRM - CUSTOMER RELATIONSHIP MANAGEMENT ( MÜŞTERİ İLİŞKİLERİ YÖNETİMİ)

CRM yöntemleri müşteriler ile kurulacak iletişimi daha farklı ve farkındalıklı açıdan ele almamızı sağlar.

## CRM Kavramları:

1. Müşteri Yaşam Döngüsü Optimizasyonları (customer lifecycle / journey / funnel)
2. İletişim ( dil, renk, görseller, kampanyalar)
3. Müşteri edinme / bulma yolları
4. Müşteri elde tutma (terk) çalışmaları

Not: Pazarlama101 e göre yeni müşteriler bulmak eldeki müşterileri tutmaktan daha maliyetlidir! Çünkü yeni müşteriler yeni çabalar ve de yeni maliyetler harcanarak elde edilir.

1. Çapraz (cross sell), Üst Satış (Upsell): Örneğin hamburger in yanında patetes cipsi ve kola satmak cross sell, hamburger in büyük boyunu satmak ise upsell olarak adlandırılır!
2. Müşteri Segmentasyon Çalışmaları: Bir marka müşterilerini segmentlere ayırarak her segmentteki müşterilerine farklı şekilde hitap eder ! Kaynağın sınırlı ve yapılacak işin fazla olması nedeniyle müşteriler alt segmentlere ayırılır.

Not: Yukarıdaki kavramların hepsi kural tabanlı ya da geleneksel yöntemler ile gerçekleştirilebilmektedir. Fakat CRM Analitiğinin temel amacı(veri bilimcinin CRM Analitiği yapma amacı), **veriye dayalı olarak** tüm Müşteri ilişkileri sürecini daha verimli hale getirmektektir.

**CRM Analitiği** : CRM kapsamındaki çalışmaların veriye dayalı, veri odaklı gerçekleştirilme çabasını barındıran bir veri analitiği, veri bilimi uygulama alanıdır!

CRM Analitiği şirketlere strateji geliştirme imkanı da sağlamaktadır.(daha fazla müşteri, kazanç, daha az çabayla daha fazla kaynak oluşturmak gibi stratejiler)

CRM Analitiği hemen hemen her şirkette vardır, tek başına CRM de iyi olmak bir çok pozisyonun kapılarını açacaktır!

## Key Performance Indicators - KPIs ( Temel Performans Göstergeleri)

Şirket, departman ya da çalışanların performanslarını değerlendirmek için kullanılan matematiksel göstergelerdir.

### **KPI Örnekleri:**

1. Customer Acquisition Rate (Müşteri Kazanma Oranı): Belirli bir zamanda kazanılan müşteri yüzdesi
2. Customer Retention Rate (Müşteri Elde Tutma Oranı):

Not: Müşteri ikinci alışverişini yaptıktan sonra bizim müşterimiz gibi kabul edilir !

1. Customer Churn Rate (Müşteri Terk Oranı):
2. Conversion Rate ( Dönüşüm Oranı): Örneğin bir ilan sitesinde ilan verildi, ilanı 1000 kişi gördü ve 10 kişi tıkladıysa Conversion Rate = 10 / 1000 olur. Tıklayan 10 kişi arasından 1 kişi satın aldıysa bunun dönüşüm oranı(conversion Rate) = 1 / 10 olur. Conversion Rate ile amaç belirli bir süreçteki etkileşimleri matematiksel olarak takip etme çabasıdır.
3. Growth Rate (Büyüme Oranı)

KPIs(Temel Performans Göstergeleri) örnekleri çok fazlasıyla artırılabilir. Burdaki temel amaç ilgili göstergeleri matematiksel anlamda takip etmeye çalışmaktır!

## Kohort Analizi ( Analysis of Cohort):

Cohort: Kelime anlamı topluluk. Genel manada, ortak özellikliklere sahip bir grup insan.

Cohort Analizi: Ortak özelliklere sahip bir grup insanın, davranışlarının **(zamana göre)** analizi. Ortak özellikler; churn rate(markayı terk eden insanlar), ortak özellikli insanların analizi: zamana göre churn / terk oranlarının incelenmesidir.

## RFM İLE MÜŞTERİ SEGMENTASYONU:

RFM - Recency, Frequency, Monetary (En sonki müşteri satın alması, müşterinin ne kadar ürün satın aldığı, müşterinin bıraktığı para)

RFM analizi müşteri segmentasyonu için kullanılan bir tekniktir. Müşterilerin satın alma alışkanlıkları üzerinden gruplara ayrılması ve bu gruplar üzerinde strateji geliştirilmesini sağlar.

CRM çalışmaları kapsamında veriye dayalı aksiyon alınmasını sağlar.

RFM Metrikleri 🡪 RFM skorları 🡪 Müşteri Segmentasyonu

### RFM Metrikleri:

Recency (Yenilik): Müşterinin bizden en son ne zaman alışveriş yaptığını gösteriyor. Ne kadar düşükse o kadar iyidir. **( today\_date – last\_purchas\_date)**

Frequency (Sıklık): Müşterinin Yaptığı toplam alışveriş / işlem sayısıdır. **(total transaction)**

Monetary (Parasal Değer): Müşterinin bize bıraktığı toplam para miktarıdır. **(total monetary)**

**NOT:** CRM çalışmalarında frequency monetary den daha önemlidir. Çünkü daha fazla sıklığı olan müşteri toplamda daha fazla getiri getirecektir.

PYTHON KODLARI:

### RFM ANALİZİ  
# RFM - Recency, Frequency, Monetary (En sonki müşteri satın alması, müşterinin ne kadar ürün satın aldığı, müşterinin bıraktığı para)  
# RFM analizi müşteri segmentasyonu için kullanılan bir tekniktir. Müşterilerin satın alma alışkanlıkları üzerinden gruplara ayrılması ve bu gruplar üzerinde strateji geliştirilmesini sağlar.  
# CRM çalışmaları kapsamında veriye dayalı aksiyon alınmasını sağlar.  
# RFM Metrikleri ◊ RFM skorları ◊ Müşteri Segmentasyonu  
# RFM Metrikleri:  
# Recency (Yenilik): Müşterinin bizden en son ne zaman alışveriş yaptığını gösteriyor. Ne kadar düşükse o kadar iyidir.  
# Frequency (Sıklık): Müşterinin Yaptığı toplam alışveriş / işlem sayısıdır.  
# Monetary (Parasal Değer): Müşterinin bize bıraktığı toplam para miktarıdır.  
# NOT: CRM çalışmalarında frequency monetary den daha önemlidir. Çünük daha fazla sıklığı olan müşteri toplamda daha fazla getiri getirecektir.  
  
  
# 1. İş Problemi:  
# 2. Veriyi Anlama (Data Understanding)  
# 3. Veri Hazırlama ( Data preparation)  
# 4. RFM Metriklerinin Hesaplanması  
# 5. RFM Sokorlarının Hesaplanması  
#6. RFM Segmentlerinin oluşturulması ve analiz edilmesi.( Creating & Analysing RFM Segments)  
#7. Tüm Sürecin Fonksiyonlaştırılması  
  
  
#1. İş Problemi:  
# Bir e ticaret şirketi müşterilerini segmetnlere ayırıp bu segmentlere göre pazarlama stratejisi belirlemek istiyor.  
# Bir e-ticaret şirketi müşterilerini segmentlere ayırıp bu segmentlere göre  
# pazarlama stratejileri belirlemek istiyor.  
  
# Veri Seti Hikayesi  
# https://archive.ics.uci.edu/ml/datasets/Online+Retail+II  
  
# Online Retail II isimli veri seti İngiltere merkezli online bir satış mağazasının  
# 01/12/2009 - 09/12/2011 tarihleri arasındaki satışlarını içeriyor.  
  
# Değişkenler  
#  
# InvoiceNo: Fatura numarası. Her işleme yani faturaya ait eşsiz numara. C ile başlıyorsa iptal edilen işlem.  
# StockCode: Ürün kodu. Her bir ürün için eşsiz numara.  
# Description: Ürün ismi  
# Quantity: Ürün adedi. Faturalardaki ürünlerden kaçar tane satıldığını ifade etmektedir.  
# InvoiceDate: Fatura tarihi ve zamanı.  
# UnitPrice: Ürün fiyatı (Sterlin cinsinden)  
# CustomerID: Eşsiz müşteri numarası  
# Country: Ülke ismi. Müşterinin yaşadığı ülke.  
  
# 2. Veriyi Anlama (Data Understanding)  
import pandas as pd, numpy as np  
pd.set\_option("display.max\_columns", None)  
# pd.set\_option("display.max\_rows", None)  
pd.set\_option("display.width", 1000)  
pd.set\_option("display.float\_format", lambda x: "%.3f" % x) # float sayıların virgülden sonra 3 basamağını da göstermesi ayarlanıyor!  
df\_ = pd.read\_excel("online\_retail\_II.xlsx", sheet\_name="Year 2009-2010")  
df = df\_.copy()  
df.head(100)  
df.shape  
df.info()  
df.isnull().sum()  
df.columns = df.columns.str.lower()  
df["description"].nunique()  
df["invoice"].nunique()  
df["description"].value\_counts().head() # kaç farklı ürün var  
df.groupby("description").agg({"quantity" : "sum"}).sort\_values("quantity", ascending=False) # en fazla hangi üründen satılmış  
df.isnull().sum()  
df["invoice"].nunique() # kaç farklı fatura kesilmiş  
#### 3. veriyi hazırlama  
  
  
df["total\_price"] = df["quantity"] \* df["price"]  
df.groupby("invoice").agg({"total\_price": "sum"}).sort\_values("total\_price", ascending=False) # hangi faturadan toplamda ne kadar kazanılmış  
df.isnull().sum()  
df.dropna(inplace=True) # verideki customerid değişkeninde eksiklik olduğundan boş değerleri sildik !  
#df.dropna(subset = ['column\_name']) # specific bir kolondaki değerleri silmek  
df.shape  
df.head()  
df = df[~df["invoice"].str.contains("C", na=False)] # contains metodunda na=False kullanılarak boş değerler False olarak algılanıyor/yazılıyor  
# df[~df["invoice"].str.contains("C", na=False)] ile fatura kısmında iptal edilen(başında C olan) kısımlar df in dışında bırakılıyor!  
df.describe().T  
  
# 4. RFM metriklerinin hesaplanması - Recency, Frequency, Monetary - calculating rfm metrics  
df.head()  
df["invoicedate"].max() #çıktı: Timestamp('2010-12-09 20:01:00') . analiz tarihinini en sonki alışveriş tarihini bulup üstüne 2 gün ekledikten sonra buluyoruz.  
import datetime as dt  
today\_date = dt.datetime(2010, 12, 11)  
type(today\_date)  
rfm = df.groupby("customer id").agg({"invoicedate": lambda date: (today\_date - date.max()).days, #Recency - bugun ile en sonki alışveriş arasındaki fark  
 "invoice" : lambda num: num.nunique(), #Kaç farklı fatura kesilmiş - frequency - kaç farklı alışveriş yapılmış  
 "total\_price": lambda total: total.sum()}) #müşterinin toplam bıraktığı para  
rfm.columns = ["recency", "frequency", "monetary"]  
rfm.head()  
rfm.describe().T  
rfm = rfm[rfm["monetary"] > 0] ## parasal değeri 0 dan büyük olanları seçiyoruz.  
  
rfm.head()  
rfm.shape  
  
#5 - rfm skorlarının hesaplanması - calculating rfm scores  
  
rfm.head()  
rfm.columns  
rfm["recency\_score"] = pd.qcut(rfm.loc[:, "recency"], 5, labels=[5, 4, 3, 2, 1])  
#qcut: quntile(çeyreklik) değerlerine göre bölme yapar. öncelikle veriyi küçükten büyüğe sıralar ve ilgili etiketleri ilgili aralığıa koyar  
# recency - müşterinin alışveriş yaptığı en sonki günlük süre- ne kadar küçükse bizim için o kadar iyi olduğundan en küçük recency  
# değerine en büyük recency skoru(5) etiket olarak verildi !!1  
rfm.head()  
rfm["monetary\_score"] = pd.qcut(rfm["monetary"], 5, labels=[1, 2, 3, 4, 5])  
# yuarıdaki labels kısmında büyük görülen parçalar büyük, küçük görülen parçalar ise küçük sayıyla yazılacka !  
  
rfm["frequency\_score"] = pd.qcut(rfm["frequency"], 5, labels=[1, 2, 3, 4, 5])  
# yukarıdaki kod ile hata alınıyor :You can drop duplicate edges by setting the 'duplicates' kwarg  
# yani 5 parçaya bölünen çeyreklik değerlerde örneğin; ilk çeyreklik ile ikinci çeyreklik değerlerinde çok sayısa aynı sayıdan var.  
#bu durumda rank metodu ile, rank(method="first") kullanılarak ilgili değişken sıralanarak çok sayıdaki değerler hangi sıralamadaysa o sıralamayı koruyarak parçalara bölünüyor.  
# yani rank(method="first") ile qcut ın ilk gördüğü değerler ilk sıralamaya alınıyor  
rfm["frequency\_score"] = pd.qcut(rfm["frequency"].rank(method="first"), 5, labels=[1, 2, 3, 4, 5])  
rfm.head()  
rfm["RFM\_score"] = rfm["recency\_score"].astype(str) + rfm["frequency\_score"].astype(str)  
  
#6. RFM segmentlerinin oluşturulması ve Analiz Edilmesi (creating and Analysing RFM segments)  
seg\_map = {  
 r'[1-2][1-2]': 'hibernating',  
 r'[1-2][3-4]': 'at\_Risk',  
 r'[1-2]5': 'cant\_loose',  
 r'3[1-2]': 'about\_to\_sleep',  
 r'33': 'need\_attention',  
 r'[3-4][4-5]': 'loyal\_customers',  
 r'41': 'promising',  
 r'51': 'new\_customers',  
 r'[4-5][2-3]': 'potential\_loyalists',  
 r'5[4-5]': 'champions'  
}  
seg\_map = {r"[1-2][1-2]" : "hibernating",  
 r"[1-2][3-4]" : "at\_risk",  
 r"[1-2]5" : "cant\_loose",  
 r"3[1-2]" : "about\_to\_sleep",  
 r"33" : "need\_attention",  
 r"[3-4][4-5]" : "loyal\_customers",  
 r"41" : "promissing",  
 r"51" : "new\_customers",  
 r"[4-5][2-3]" : "potential\_loyalists",  
 r"5[4-5]" : "champions"  
 }  
  
# r"[1-2][3-4]" : "at\_risk" r ifaredesi regex in kısaltması. [] her bir elemanı ifade ediyor. buradaki ifade ilk elamnında 1 ya da 2 geçen ve ikinci elemanında 3 ya da 4  
# geçen ifadelere at rist yazdırıyor.  
rfm["segment"] = rfm["RFM\_score"].replace(seg\_map,regex=True)  
rfm[["segment", "recency", "frequency", "monetary" ]].groupby("segment").agg(["mean", "count"])  
rfm[rfm["segment"] == "need\_attention"]  
  
new\_df = pd.DataFrame()  
  
new\_df["new\_customer\_id"] = rfm[rfm["segment"] == "new\_customers"].index  
new\_df["new\_customer\_id"] = new\_df["new\_customer\_id"].astype(int)  
new\_df.to\_csv("new\_customers.csv")  
rfm.to\_csv("rfm.csv")  
  
#### 7. TÜM SÜRECİN FONKSİYONLAŞTIRLMASI  
def create\_rfm(dataframe, csv=False):  
 # veriyi hazırlama  
 dataframe["TotalPrice"] = dataframe["Quantity"] \* dataframe["Price"]  
 dataframe.dropna(inplace=True)  
 dataframe = dataframe[~dataframe["Invoice"].str.contains("C", na=False)] # contains metodunda na=False kullanılarak boş değerler False olarak algılanıyor/yazılıyor  
 # RFM Metriklerinin Hesaplanması  
 import datetime as dt  
 today\_date = dt.datetime(2011, 12, 11)  
 rfm = df.groupby("Customer ID").agg({"InvoiceDate": lambda date: (today\_date - date.max()).days,  
 # Recency - bugun ile en sonki alışveriş arasındaki fark  
 "Invoice": lambda num: num.nunique(),  
 # Kaç farklı fatura kesilmiş - frequency - kaç farklı alışveriş yapılmış  
 "TotalPrice": lambda total: total.sum()})  
 # müşterinin toplam bıraktığı para  
 rfm.columns = ["recency", "frequency", "monetary"]  
 rfm = rfm[rfm["monetary"] > 0] ## parasal değeri 0 dan büyük olanları seçiyoruz.  
 rfm["recency\_score"] = pd.qcut(rfm.loc[:, "recency"], 5, labels=[5, 4, 3, 2, 1])  
 rfm["monetary\_score"] = pd.qcut(rfm["monetary"], 5, labels=[1, 2, 3, 4, 5])  
 rfm["frequency\_score"] = pd.qcut(rfm["frequency"].rank(method="first"), 5, labels=[1, 2, 3, 4, 5])  
 rfm["RFM\_score"] = rfm["recency\_score"].astype(str) + rfm["frequency\_score"].astype(str)  
 seg\_map = {r"[1-2][1-2]": "hibernating",  
 r"[1-2][3-4]": "at\_risk",  
 r"[1-2]5": "cant\_loose",  
 r"3[1-2]": "about\_to\_sleep",  
 r"33": "need\_attention",  
 r"[3-4][4-5]": "loyal\_customers",  
 r"41": "promissing",  
 r"51": "new\_customers",  
 r"[4-5][2-3]": "potential\_loyalists",  
 r"5[4-5]": "champions"  
 }  
 rfm["segment"] = rfm["RFM\_score"].replace(seg\_map, regex=True)  
 rfm = rfm[["segment", "recency", "frequency", "monetary"]]  
 rfm.index.astype(int)  
 if csv:  
 rfm.to\_csv("rfm.csv")  
 return rfm  
new\_rfm = create\_rfm(df, csv=True)

## MÜŞTERİ YAŞAM BOYU DEĞERİ:

### Customer Life Time Value - CLTV (Müşteri Yaşam Boyu Değeri):

Bir müşterinin bir şirketle kurduğu ilişki-iletişim süresince bu şirkete kazandıracağı parasal değerdir.

Bir müşterinin işletme için üreteceği bütün gelirlerin beklenen değeri/toplamı

### Nasıl Hesaplanır ?

Literatürde birçok yolu olduğu söylenmektedir. Temel olarak ise:

**CLTV(Customer LifeTime Value)= (Customer Value / Churn Rate) \* Profit Margin**

**CV (Customer Value) = satın alma başına ortalama kazanç \* satın alma sayısı**

**CV (Customer Value) = Average Order Value \* Purchase Frequency**

**Not: Customer Value Çok önelidir.**

**Average Order Value = Total Price / Total Transaction**

**Purchase Frequency = Total Transaction / Total Number of Customers**

**Churn Rate = 1 – Repat rate**

**Müşteri terk oranı = 1 – müşteri elde tutma oranı(repeat rate)**

**Müşteri elde tutma oranı (repeat rate) = birden fazla alışveriş yapan müşteriler / tüm müşteriler**

**Profit Margin = Total Price \* 0,10**

**Transaction: İşlem(Bizim için fatura kesilme işlemi)**

**Sonuç olarak: Her bir müşteri için hesaplancak olan CLTV değerlerine göre bir sıralama yapıldığında ve CLTV değerlerine göre belirli noktalardan bölme işlemi yapılarak gruplar oluşturulduğunda müşterimiz Segmentlere ayrılacaktır!**

import pandas as pd, numpy as np  
pd.set\_option("display.max\_columns", None)  
# pd.set\_option("display.max\_rows", None)  
pd.set\_option("display.width", 1000)  
pd.set\_option("display.float\_format", lambda x: "%.5f" % x) # float sayıların virgülden sonra 3 basamağını da göstermesi ayarlanıyor!  
df\_ = pd.read\_excel("online\_retail\_II.xlsx", sheet\_name="Year 2009-2010")  
df = df\_.copy()  
df.head()  
# 1. Veriyi Hazırlama  
from sklearn.preprocessing import MinMaxScaler  
df = df[~df["Invoice"].str.contains("C", na=False)]  
df.dropna(subset= "Customer ID", inplace=True) # sadece customerid kolonundaki boşları çıkardı.  
df.dropna(inplace=True)  
df.shape  
df = df[df["Quantity"] > 0]  
df.describe().T  
  
df["TotalPrice"] = df["Quantity"] \* df["Price"]  
df.head()  
  
cltv\_c = df.groupby("Customer ID").agg({"Invoice": lambda x: x.nunique(),  
 "Quantity": lambda x: x.sum(),  
 "TotalPrice": lambda x: x.sum()})  
#cltv\_c alternatifi:  
# cltv\_c = df.groupby("Customer ID").agg({"Invoice": lambda x: x.nunique(),  
# "Quantity": "sum",  
# "TotalPrice": "sum"})  
cltv\_c.columns = ["total\_transaction", "total\_unit", "total\_price"]  
  
  
# 2. Average Order Value(Oratalama Sipariş Değeri) = Total Price / Total Transaction  
cltv\_c.head()  
cltv\_c["average\_order\_value"] = cltv\_c["total\_price"] / cltv\_c["total\_transaction"]  
  
#3. Purchaase Frequency (Satın Alma Sıklığı) = Total Transaction / Total Number of Customers  
  
cltv\_c["purchase\_frequency"] = cltv\_c["total\_transaction"] / cltv\_c.shape[0]  
  
  
#4. Tekrarlama Oranı & Müşteri Terki(kaybetme) Oranı (Repeat rate & churn Rate) = birden fazla alışveriş yapan müşteriler / tüm müşteriler  
  
repeat\_rate = cltv\_c[cltv\_c["total\_transaction"] > 1].shape[0] / cltv\_c.shape[0]  
churn\_rate = 1 - repeat\_rate  
  
# profit margin - kar margin. Profit Margin = Total Price \* 0,10. not: buradaki 0.10 değeri şirket tarafından belirlenmesi gereken bir değerdir.  
cltv\_c["profit\_margin"] = cltv\_c["total\_price"] \* 0.10  
  
#customer value - müşteri değeri (Customer Value) = Average Order Value \* Purchase Frequency  
cltv\_c["customer\_value"] = cltv\_c["average\_order\_value"] \* cltv\_c["purchase\_frequency"]  
  
# customer lifetime value - müşteri yaşamboyu değeri = Customer Value / Churn Rate \* Profit Margin  
  
cltv\_c["cltv"] = (cltv\_c["customer\_value"] / churn\_rate) \* cltv\_c["profit\_margin"]  
cltv\_c.sort\_values(by="cltv", ascending=False)  
  
#müşterilerin segmetlere ayrılması  
cltv\_c.sort\_values(by="cltv", ascending=False)  
cltv\_c["segment"] = pd.qcut(cltv\_c["cltv"], 4, labels=["D", "C", "B", "A"])  
cltv\_c.groupby("segment").agg({"count", "mean", "sum"})  
cltv\_c.to\_csv("cltv\_c.csv")  
  
### Tüm işlemlerin fonksitonlaştırması  
def create\_cltv\_c(dataframe, profit=0.10):  
 #veri hazırlama  
 dataframe.dropna(inplace=True)  
 dataframe = dataframe[~dataframe["Invoice"].str.contains("C", na=False)]  
 dataframe = dataframe[dataframe["Quantity"] > 0]  
 dataframe["TotalPrice"] = dataframe["Quantity"] \* dataframe["Price"]  
 cltv\_c = dataframe.groupby("Customer ID").agg({"Invoice": lambda x: x.nunique(),  
 "Quantity": lambda x: x.sum(),  
 "TotalPrice": lambda x: x.sum()})  
 cltv\_c.columns = ["total\_transaction", "total\_unit", "total\_price"]  
 # 2. Average Order Value(Oratalama Sipariş Değeri) = Total Price / Total Transaction  
 cltv\_c["average\_order\_value"] = cltv\_c["total\_price"] / cltv\_c["total\_transaction"]  
 # 3. Purchaase Frequency (Satın Alma Sıklığı) = Total Transaction / Total Number of Customers  
 cltv\_c["purchase\_frequency"] = cltv\_c["total\_transaction"] / cltv\_c.shape[0]  
 # 4. Tekrarlama Oranı & Müşteri Terki(kaybetme) Oranı (Repeat rate & churn Rate) = birden fazla alışveriş yapan müşteriler / tüm müşteriler  
 repeat\_rate = cltv\_c[cltv\_c["total\_transaction"] > 1].shape[0] / cltv\_c.shape[0]  
 churn\_rate = 1 - repeat\_rate  
 # profit margin - kar margin. Profit Margin = Total Price \* 0,10. not: buradaki 0.10 değeri şirket tarafından belirlenmesi gereken bir değerdir.  
 cltv\_c["profit\_margin"] = cltv\_c["total\_price"] \* profit  
 # customer value - müşteri değeri (Customer Value) = Average Order Value \* Purchase Frequency  
 cltv\_c["customer\_value"] = cltv\_c["average\_order\_value"] \* cltv\_c["purchase\_frequency"]  
 # customer lifetime value - müşteri yaşamboyu değeri = Customer Value / Churn Rate \* Profit Margin  
 cltv\_c["cltv"] = (cltv\_c["customer\_value"] / churn\_rate) \* cltv\_c["profit\_margin"]  
 #segmenlerin oluştuurlması  
 cltv\_c["segment"] = pd.qcut(cltv\_c["cltv"], 4, labels=["D", "C", "B", "A"])  
 return cltv\_c  
  
  
create\_cltv\_c(df)

## Müşteri Yaşam Boyu Tahmini (BG-NBG ve GAMMA – GAMMA Modelleri):

Zaman Projeksiyonlu olasılıksal Life Time Value Tahmini: Yani müşterinin şirketle kurduğu ilişki dönemi boyunca toplamda ne kadar para getireceğinin tahminini yapıyoruz.

**CV (Customer Value) = satın alma başına ortalama kazanç \* satın alma sayısı**

**Not: Yukarıdaki satın alma başına ortalama kazanç müşterinin toplam yaptığı alışverişlerden elde edilirken satın alma sayısı belirli bir periyot için düşünülebilir. Örneğin bir müşteri toplamda 30 birimlik 10 farklı alışveriş yaptıysa bu müşterinin aylık bazda getireceği alışveriş bu satın alma başına gerçekleştirilen \* aylık ortalama alışveriş sayısı ile bulunuyor.**

**Olasılıksal bazda bir müşterinin şirketle ilişkisi boyunca ne kadar getireceğini hesaplamak için;**

**CLTV = Conditional Expected Number of Transaction \* Conditional Expected Average Profit**

**CV (Customer Value) = Purchase Frequency \* Average Order Value**

**Not: Purchase = Transaction**

**Conditional Expected Number of Transaction: Bütün kitlenin satın alma davranışını bir olasılık dağılımıyla modellendikten sonra, bu genel satın alma davranışını kişi özelinde(conditional) biçimlendirecek şekilde kullanarak her bir müşteri için beklenen satın alma / işlem sayıları tahmin edilir.**

**Conditional Expected Average Profit: Bütün kitlenin Average Profit değerini olasılıksal olarak modelledikten sonra, bu modele kişi özelliklerini vererek, kişilerin özelinde Conditional Expected Average Profit değerleri hesaplanır.**

**CV (Customer Value) = Purchase Frequency \* Average Order Value**

**CLTV = Conditional Expected Number of Transaction \* Conditional Expected Average Profit**

**Conditional Expected Number of Transaction: BG-NBD modeli ile hesaplanıyor**

**Conditional Expected Average Profit: Gamma Gamma Submodel ile hesaplanıyor.**

**CLTV = BG-NBD modeli \* Gamma Gamma Submodel**

### BG / NBD ( Beta Geometric / Negative Binomial Distribution) ile Expected Number of Transaction: ( Expected Number of Transaction = Expected Sales Forecasting)

**CLTV = Conditional Expected Number of Transaction \*** Conditional Expected Average Profit

**CLTV = BG-NBD modeli \*** Gamma Gamma Submodel

**Expected:** Bir rastsal değişkenin beklenen değerini(ortalamasını) ifade eder.

**Rastsal Değişken:** Değerlerini bir deneyin sonuçlarından alan değişkene rastsal değişken denir. Bir değişkenin belirli bir olasılık dağılımı izlediğini varsaydığımızda bu değişkenin ortalamasına rastsal değişken denir.

**BG-NBD Modeli literatürde buy till you die olarak da tanımlanmaktadır. Buy till you die demek, öncelikle satın alma davranışı sergilemek ve sonrasında bu süreci sonlandırarak drop, churn olmak (till you die) anlamına gelir**.

BG-NBD Modeli, Expected Number of Transaction için iki süreci olasılıksal olarak modeller.

1. Transaction Process ( Buy)
2. Dropout Process – inaktif olma süreci (Till you die)

#### Transaction Process(Buy):

* Alive(Canlı) olduğu sürece, belirli bir zaman periyodunda, bir müşteri tarafından gerçekleştirilecek işlem sayısı transaction rate parametresi ile possion dağılır.
* Bir müşteri Alive olduğu sürece (Dropout olmadığı sürece) kendi transaction rate’i etrafında rastgele satın alma yapmaya devam edecektir.
* Transaction rate’ler her bir müşteriye göre değişir ve tüm kitle için gamma dağılır (r,a).

#### Dropout Process (Till you die):

* Her bir müşterinin p olasılığı ile dropout rate (dropout probability)’i vardır.
* Bir müşteri alışveriş yaptıkan sonra belirli bir olasılıkla drop olur.
* Dropout rate’ler her bir müşteriye göre değişri ve tüm kitle için beta dağılır(a, b).

### Gamma Gamma Submodel

Bir müşterinin işlem başına ortalama ne kadar kar getirebileceğini tahmin etmek için kullanılır.

**CLTV =** Conditional Expected Number of Transaction **\* Conditional Expected Average Profit**

**CLTV =** BG-NBD modeli **\* Gamma Gamma Submodel**

* Bir müşterinin işlemlerinin parasal değeri (monetary) transaction value’ların ortalaması etrafında rastgele dağılır.
* Ortalama transaction value, zaman içinde kullanıcılar arasında değişebilir fakat tek bir kullanıcı için değişmez. ?
* Ortalama transaction value tüm müşteriler arasında gamma dağılır.

conditional\_expected\_number\_of\_purchases\_up\_to\_time = **=** Conditional Expected Number of Transaction

**# NOT:** **BG-NDB (buy till you die) teorisine göre düzenli ortalama bir alışveriş yapan bir müşteri eğer churn/dropout olmadıysa, müşterinin recency değeri arttıkça o müşterinin satın alma ihtimali artıyordur.**

**# ÇOK ÖNEMLİ NOT: cltv hesaplanırken frequency değerleri integer olması gerekmektedir.Bu nedenle alt ve üst limitlerini round() ile yuvarlayınız.**

**# CRM CASE STUDIES NOTLARI:**

1. **1 den fazla sütunun datetime a çevirmek için:**

data[cols] = data[cols].apply(pd.to\_datetime, errors='coerce')

With errors='coerce', invalid dates are assigned to NaT

1. **df["last\_day\_to\_firs\_day"] = (df["last\_date"] - df["firs\_date"]).apply(lambda x:x.days) ile**

**df["last\_day\_to\_firs\_day"] = (df["last\_date"] - df["firs\_date"]).dt.days aynı kodlardır !**

**#BG-NBD(BetaGeoFitter) ve Gamma Gamma modellerinin (GammaGammaFitter) İstediği veri formatlarının hazırlanması**

**RFM METRİKLERİ:**

**RFM Metriklerinin Hesaplanması**

**# Recency (Yenilik): Müşterinin bizden en son ne zaman alışveriş yaptığını gösteriyor. Ne kadar düşükse o kadar iyidir.**

**# Frequency (Sıklık): Müşterinin Yaptığı toplam alışveriş / işlem sayısıdır.**

**# Monetary (Parasal Değer): Müşterinin bize bıraktığı toplam para miktarıdır.**

**CLTV METRİKLERİ:**

**# recency: Son satın alma üzerinden geçen zaman. Haftalık. (kullanıcı özelinde) - müşterinin kendi içinde son satın alma zamanı - ilk satın alma zamanını ifade eder**

**# T: Müşterinin yaşı. Haftalık. (analiz tarihinden ne kadar süre önce ilk satın alma yapılmış)**

**# frequency: tekrar eden toplam satın alma sayısı (frequency>1)**

**# monetary: satın alma başına ortalama kazanç**

**ÇOK ÇOK ÖNEMLİ NOT: FORMÜLDEN HESAPLANAN CLTV DEĞERİ VERİ SETİNİN SÜRESİ BOYUNCA OLAN ALIŞVERİŞLERİ KAPSIYOR. YANİ ŞU ANA KADAR OLANKİ CLTV DEĞERİ HESAPLANMIŞ OLUYOR.**

**CLTV(Customer LifeTime Value)= (Customer Value / Churn Rate) \* Profit Margin**

**CV (Customer Value) = satın alma başına ortalama kazanç \* satın alma sayısı**

**CV (Customer Value) = Average Order Value \* Purchase Frequency**

**Not: Customer Value Çok önelidir.**

**Average Order Value = Total Price / Total Transaction**

**Purchase Frequency = Total Transaction / Total Number of Customers**

**Churn Rate = 1 – Repat rate**

**Müşteri terk oranı = 1 – müşteri elde tutma oranı(repeat rate)**

**Müşteri elde tutma oranı (repeat rate) = birden fazla alışveriş yapan müşteriler / tüm müşteriler**

**Profit Margin = Total Price \* 0,10**

**Transaction: İşlem(Bizim için fatura kesilme işlemi)**

**BG-NDB VE GAMMA GAMMA SUB MODEL İLE DE ELİMİZDEKİ VERİ SETİYLE MÜŞTERİLERİN GELECEKTEKİ CLTV DEĞERLERİNİ OLASIKLIKSAL OLARAK TAHMİN EDİYORUZ.**

**CLTV = Conditional Expected Number of Transaction \* Conditional Expected Average Profit**

**Conditional Expected Number of Transaction: BG-NBD modeli ile hesaplanıyor**

**Conditional Expected Average Profit: Gamma Gamma Submodel ile hesaplanıyor.**

**CLTV = BG-NBD modeli \* Gamma Gamma Submodel**

**NOT: MÜŞTERİLERİ SEGMENTLERE AYIRMAK SINIRLI PAZARLAMA KAYNAKLARINDAN DOLAYI HER ŞİRKET İÇİN TEMEL BİR İHTİAÇTIR Kİ, ŞİRKET HER MÜŞTERİSİNE FARKLI OLARAK DAVRANACAKTIR**

**NOT:** **QUANTILE HESAPLANIRKEN VERİ KÜÇÜKTEN BÜYÜĞE SIRALANIP KAÇINCI ÇEVREKLİK DEĞER İSTENİYORSA O HESAPLANIYOR. % 50 QUANTILE DİZİNİN MEDYANI OLUYOR !**

**NOT:** DİZİNİN ORTALAMA DEĞERİ MEDYAN’DAN BÜYÜKSE AYKIRI DEĞER BARINDIRIYOR.

**NOT:** RFM METRİKLERİNE GÖRE BELİRTİLEN SEG-MAP KISMIŞİRKET ÖZELİNDE YORUMLARA GÖRE DEĞİŞEBİLİR.

**seg\_map = {**

**r'[1-2][1-2]': 'hibernating',**

**r'[1-2][3-4]': 'at\_Risk',**

**r'[1-2]5': 'cant\_loose',**

**r'3[1-2]': 'about\_to\_sleep',**

**r'33': 'need\_attention',**

**r'[3-4][4-5]': 'loyal\_customers',**

**r'41': 'promising',**

**r'51': 'new\_customers',**

**r'[4-5][2-3]': 'potential\_loyalists',**

**r'5[4-5]': 'champions'**

**}**