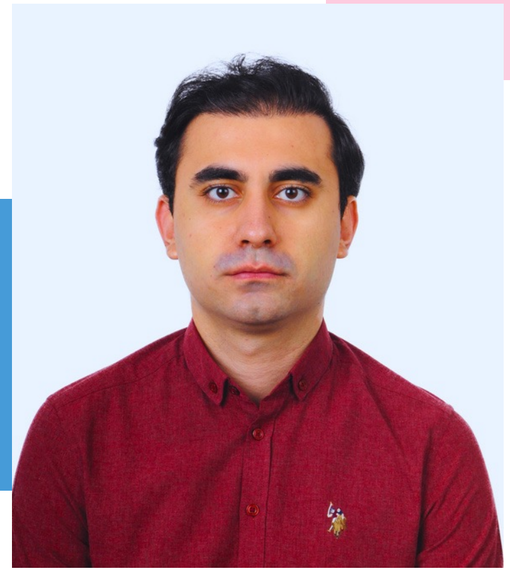
PySpark ile Müşteri Segmentasyonu

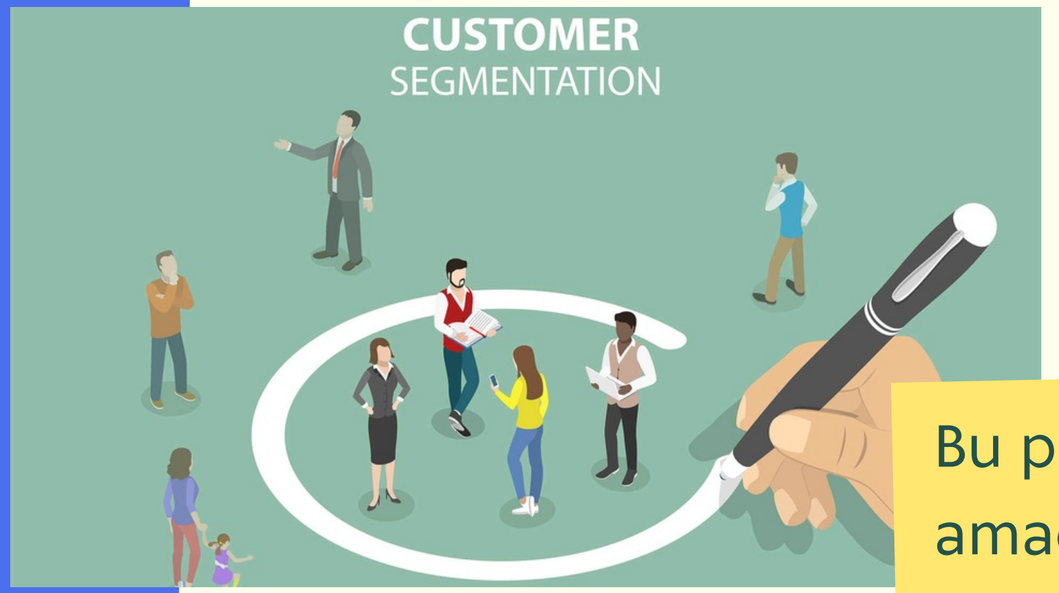


Mustafa Serdar Konca

İÇINDEKILER

İÇINDEKILER

Proje Tanıtımı



Bu proje, müşteri segmentasyonunu belirlemek amacıyla **PySpark** kullanarak **veri analizi** ve **modelleme** yapmayı hedeflemektedir. Müşteri segmentasyonu, müşterilerin davranışlarını ve özelliklerini anlamak için kullanılır ve pazarlama stratejilerini optimize etmekte önemli bir rol oynar.

Veri Seti Tanıtımı

Kullanılan veri seti, bir **e-ticaret** sitesinin **müşteri işlem kayıtları**nı içermektedir. **Veri seti, fatura numarası, stok kodu, ürün açıklaması, miktar, fatura tarihi, birim fiyat, müşteri kimliği ve ülke** gibi özellikler içermektedir.

KÜTÜPHANELER

```
:
 1 from pyspark.ml.clustering import KMeans
 2 from pyspark.ml.evaluation import ClusteringEvaluator
 3 import numpy as np
 4 from pyspark.sql import SparkSession
 5 from pyspark.sql.functions import *
 6 from pyspark.sql.types import *
 7 from pyspark.ml.feature import VectorAssembler
 8 from pyspark.ml.feature import StandardScaler
 9 import pandas as pd
10 import pylab as pl
11 import matplotlib.pyplot as plt
12 import seaborn as sns
13
14 import warnings
15 warnings.filterwarnings('ignore')
```

- Gerekli kütüphanelerimiz ekliyoruz. SparkSession, Spark'taki tüm işlevlere bir giriş noktasıdır ve PySpark'ta bir veri çerçevesi oluşturmak istiyorsanız gereklidir.

SPARKSESSION ILE BAĞLANTI KURMA


```
spark = SparkSession.builder.appName("Final_Odevi").getOrCreate()
```

- Bu kod, PySpark'i kullanarak "spark" adlı bir SparkSession nesnesi oluşturur.
  - appName parametresi, uygulamanın adını "Final_Odevi" olarak ayarlar.
  - SparkSession.builder, yığın dışı belleği etkinleştirir ve yığın dışı belleğin boyutunu 10 gigabayt olarak ayarlar.
  - Son olarak, getOrCreate() yöntemi mevcut bir SparkSession döndürür veya yoksa yeni bir tane oluşturur.

VERI SETININ YÜKLENMESI

```
1 df = spark.read.csv('eCommerce.csv', header=True, escape="\"")
```

- Csv dosyasındaki virgülleri avoid edebilmek için bir escape karakteri tanımladık ve verimizi okuduk.

VERI ÖNIZLEMESI

```
df.show(5,0)
```

```
+---------+---------+--------------------------------+--------+-----------+---------+----------+--------------+
|InvoiceNo|StockCode|Description                     |Quantity|InvoiceDate |UnitPrice|CustomerID|Country       |
+---------+---------+--------------------------------+--------+-----------+---------+----------+--------------+
|536365   |85123A   |WHITE HANGING HEART T-LIGHT HOLDER |6       |12/1/10 8:26|2.55     |17850     |United Kingdom|
|536365   |71053    |WHITE METAL LANTERN             |6       |12/1/10 8:26|3.39     |17850     |United Kingdom|
|536365   |84406B   |CREAM CUPID HEARTS COAT HANGER  |8       |12/1/10 8:26|2.75     |17850     |United Kingdom|
|536365   |84029G   |KNITTED UNION FLAG HOT WATER BOTTLE|6       |12/1/10 8:26|3.39     |17850     |United Kingdom|
|536365   |84029E   |RED WOOLLY HOTTIE WHITE HEART.  |6       |12/1/10 8:26|3.39     |17850     |United Kingdom|
+---------+---------+--------------------------------+--------+-----------+---------+----------+--------------+
only showing top 5 rows
```

- İkinci 0 argümanı, görüntülenen sütunları kesmemesi içindir, 0'a ayarlayarak, tüm sütunlar kesilmeden görüntülenecektir.

VERI AÇIKLAMASI

- Veri çerçevesi **8** feature'dan oluşur:
  - **InvoiceNo**: Her müşteri faturasının benzersiz tanımlayıcısı.
  - **StockCode**: Stoktaki her öğenin benzersiz tanımlayıcısı.
  - **Description**: Müşteri tarafından satın alınan ürün.
  - **Quantity**: Bir müşteri tarafından tek bir faturada satın alınan her bir öğenin numarası.
  - **InvoiceDate**: Satın alma tarihi.
  - **UnitPrice**: Her öğeden bir birimin fiyatı.
  - **CustomerID**: Her kullanıcıya atanan benzersiz tanımlayıcı.
  - **Country**: Satın alma işleminin yapıldığı ülke

VERI SETININ BOYUTU VE MÜŞTERI SAYISI

Adım 3: EDA (Keşifçi Veri Analizi)

Artık bu veri kümesinde bulunan değişkenleri gördüğümüze göre, bu veri noktalarını daha iyi anlamak için bazı keşifsel veri analizi yapalım:

Veri çerçevesindeki satır sayısını sayarak başlayalım:

```
1 df.count()
```

```
2500
```

Veri çerçevesinde kaç farklı müşteri var?

```
1 df.select('CustomerID').distinct().count()
```

```
95
```

- Verimiz **iki bin beş yüz satır**dan oluşmakta ve **doksan beş** farklı müşteriye ait kayıt bulunmakta.

ÜLKELERE GÖRE MÜŞTERI DAĞILIMI

```
1 df.groupBy('Country').agg(countDistinct('CustomerID').alias('country_count')).show()
```

```
+--------------+-------------+
|       Country|country_count|
+--------------+-------------+
|       Germany|            2|
|        France|            1|
|          EIRE|            1|
|        Norway|            1|
|     Australia|            1|
|United Kingdom|           88|
|   Netherlands|            1|
+--------------+-------------+
```

- agg() yöntemi daha sonra her gruba bir toplama işlevi uygulamak için kullanılır.
- countDistinct() işlevi her gruptaki farklı CustomerID değerlerinin sayısını saymak için kullanılır.
- alias() yöntemi, ortaya çıkan sütunu country_count olarak yeniden adlandırmak için kullanılır.
- Son olarak, ortaya çıkan DataFrame'i tablo biçiminde görüntülemek için show() yöntemi kullanılır.

Bu, kod sayesinde DataFrame'in Ülke sütununda her ülke için farklı müşteri sayılarını görebileceğiz.

ÜLKELERE GÖRE SIRALI MÜŞTERI DAĞILIMI

```
1 df.groupBy('Country').agg(countDistinct('CustomerID').alias('country_count')).orderBy(desc('country_count')).sho
```

```
+--------------+-------------+
|       Country|country_count|
+--------------+-------------+
|United Kingdom|           88|
|       Germany|            2|
|        France|            1|
|          EIRE|            1|
|        Norway|            1|
|     Australia|            1|
|   Netherlands|            1|
+--------------+-------------+
```

- Verideki satın alımların neredeyse tamamı Birleşik Krallık'tan yapılmış.

Yukarıdaki tablodaki verilerin satın alma sırasına göre göstermek için **orderBy()** yan tümcesini dahil edebiliriz

ZAMAN SERISI ANALIZI - MAKSIMUM TARIH

- **E-ticaret platformunda bir müşteri tarafından en son satın alma ne zaman yapıldı?**

Platformda en son satın alma işleminin ne zaman yapıldığını bulmak için, "InvoiceDate" sütununu bir zaman damgası biçimine dönüştürmemiz ve Pyspark'taki max() işlevini kullanmamız gerekiyor:

```
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
df = df.withColumn('date',to_timestamp("InvoiceDate", 'yy/MM/dd HH:mm'))
df.select(max("date")).show()
```

```
+-------------------+
|          max(date)|
+-------------------+
|2012-01-10 17:06:00|
+-------------------+
```

ZAMAN SERISI ANALIZI - MAKSIMUM TARIH

- İlk satır, **spark.sql.legacy.timeParserPolicy** yapılandırma özelliğini LEGACY olarak ayarlar. Bu özellik, Spark'ın SQL sorgularında zaman damgalarını nasıl ayrıştırdığını belirler. LEGACY olarak ayarlamak, Spark'ın daha yumuşak olan ve daha geniş bir zaman damgası biçimi yelpazesini işleyebilen eski zaman damgası ayrıştırma davranışını kullanacağı anlamına gelir.

- İkinci satır, DataFrame df'de date adında yeni bir sütun oluşturur. Bu sütundaki değerler, df'deki InvoiceDate sütununa to_timestamp işlevi uygulanarak oluşturulur. to_timestamp için ikinci argüman, InvoiceDate'deki zaman damgası dizesinin biçimini belirtir. Bu durumda, biçim yy/MM/dd HH:mm'dir, bu da zaman damgası dizesinin iki basamaklı bir yıl (yy), ardından iki basamaklı bir ay (MM), iki basamaklı bir gün (g), iki basamaklı bir saat (HH) son olarak iki basamaklı bir dakikaya (mm) olması gerektiği anlamına gelir.

- Üçüncü satır, tarih sütunundaki maksimum değeri seçer ve show() yöntemini kullanarak görüntüler.

Bu satır, InvoiceDate sütununda en son tarihi etkili bir şekilde bulur ve insan tarafından okunabilir bir biçimde görüntüler.

ZAMAN SERISI ANALIZI - MINIMUM TARIH

- **E-ticaret platformunda bir müşteri tarafından en erken satın alma ne zaman yapıldı?**
- Bir önceki yaptığımıza işleme benzer şekilde, min() işlevi en erken satın alma tarihini ve saatini bulmak için kullanılabilir.

```
1 df.select(min("date")).show()
```

```
+-------------------+
|          min(date)|
+-------------------+
|2012-01-10 08:26:00|
+-------------------+
```

- select() yöntemi, DataFrame'den belirli bir sütun veya sütun kümesini seçmek için kullanılır.
- "date" sütununun minimum değerini bulmak için min() işlevi kullanılır.
- Daha sonra işlemin sonucunu görüntülemek için show() yöntemi kullanılır.

VERI ÖN-IŞLEME

- Veri kümemizden, her kullanıcının satın alma davranışına göre birden fazla müşteri segmenti oluşturmamız gerekiyor.

- Bu veri kümesindeki değişkenler, müşteri segmentasyon modelinin kolayca anlayabileceği bir biçimde değiller. Bu öznitelikler bireysel olarak bize maalesef müşteri satın alma davranışı hakkında pek bir şey söyleyemiyor.

- Bu nedenle, mevcut değişkenler üzerinden üç yeni feature yaratacağız. - **recency, frequency, and monetary value (RFM)**.

RFM, bir müşterinin değerini aşağıdakilere göre değerlendirmek için pazarlamada yaygın olarak kullanılır:

- **Recency**: Her müşteri ne kadar zaman önce bir satın alma işlemi yaptı?
- **Frequency**: Ne sıklıkla bir şey satın aldılar?
- **Monetary Value**: Alışveriş yaparken ortalama olarak ne kadar para harcıyorlar?

Şimdi yukarıdaki değişkenleri oluşturmak için veri çerçevesini işleyeceğiz.

VERI ÖN-IŞLEME - RECENCY

```
df = df.withColumn("from_date", lit("12/1/10 08:26"))
df = df.withColumn('from_date',to_timestamp("from_date", 'yy/MM/dd HH:mm'))

df2 = df.withColumn('recency',col("date").cast("long") - col('from_date').cast("long"))
```

```
df2 = df2.join(df2.groupBy('CustomerID').agg(max('recency').alias('recency')),on='recency',how='leftsemi')
```

- İlk olarak, platformda bir satın alma işleminin yapıldığı en son tarih ve saati sonluk değeri olarak hesaplayalım. Bu işlemi iki adımda gerçekleştirebiliriz:

- Her müşteriye bir sonluk puanı atayalım. Veri çerçevesindeki her tarihi en erken tarihten itibaren çıkaracağız. Bu bize bir müşterinin veri çerçevesinde ne kadar zamanda yer aldığını söyleyecek. 0 değeri, en erken tarihte satın alma işlemi yapılan kişiye atanacağından, en düşük sonluğu göstermiş olacak.

VERI ÖN-IŞLEME - FREQUENCY

```
1 df_freq = df2.groupBy('CustomerID').agg(count('InvoiceDate').alias('frequency'))
```

```
1 df_freq.show(5,0)
```

```
+----------+---------+
|CustomerID|frequency|
+----------+---------+
|16250     |14       |
|15100     |1        |
|13065     |14       |
|12838     |59       |
|15350     |5        |
+----------+---------+
only showing top 5 rows
```

```
1 df3 = df2.join(df_freq,on='CustomerID',how='inner')
```

- Şimdi bir müşterinin platformda ne sıklıkta bir şey satın aldığını hesaplayalım. Bunu yapmak için, her müşteriyi gruplamamız ve satın aldıkları ürün sayılarına bakmamız gerekiyor.

- Ve ardından tüm bu yaptığımız işlemleri df2 veri çerçevemizde toplayalım.

VERI ÖN-IŞLEME - MONETARY VALUE

```
m_val = df3.withColumn('TotalAmount',col("Quantity") * col("UnitPrice"))
```

```
m_val = m_val.groupBy('CustomerID').agg(sum('TotalAmount').alias('monetary_value'))
```

```
finaldf = m_val.join(df3,on='CustomerID',how='inner')
```

```
finaldf = finaldf.select(['recency','frequency','monetary_value','CustomerID']).distinct()
```

- Son olarak, her müşterinin harcadığı toplam tutarı - parasal değeri hesaplayalım. Bunu yapmak için iki adım izleyebiliriz:

- I) Her satın alma işleminde harcanan toplam tutarı bulalım: Her bir müşterinin harcanan toplam tutarını elde etmek için "Miktar"ı "Birim Fiyatı" ile çarpmamız gerekir.

- ii) Her müşteri tarafından harcanan toplam tutarı bulalım: Genel olarak her bir müşterinin harcadığı toplam tutarı bulmak için, Müşteri Kimliği sütununa göre gruplamamız ve harcanan toplam tutarı toplamamız yeterlidir.

- Bu veri çerçevesini diğer tüm değişkenlerle birleştirelim ve artık modeli oluşturmak için gerekli tüm öznitelikleri oluşturduğumuza göre, yalnızca gerekli sütunları alalım.

STANDARTIZASYON

- Müşteri segmentasyon modelini oluşturmadan önce, tüm değişkenlerin aynı ölçekte olduğundan emin olmak için veri çerçevesini standartlaştıralım:

```
assemble=VectorAssembler(inputCols=[
    'recency','frequency','monetary_value'
], outputCol='features')

assembled_data=assemble.transform(finaldf)

scale=StandardScaler(inputCol='features',outputCol='standardized')
data_scale=scale.fit(assembled_data)
data_scale_output=data_scale.transform(assembled_data)
```

- Üç giriş sütununu (recency, frequency ve monetary_value) alan ve features adı verilen tek bir sütun çıkaran assemble adlı bir VectorAssembler nesnesi oluşturalım.

- Ardından, transform yöntemini kullanarak assemble dönüşümünü finaldf veri kümesine uygulayalım ve sonucu assembled_data'da saklayalım.

- Bundan sonra, assembled_data'dan özellikler sütununu alan ve standardized adlı yeni bir sütun çıkaran scale adlı bir StandardScaler nesnesi oluşturalım.

- # • Son olarak, fit edelim ve sonucu data_scale_output'ta saklayalım.

MAKINE ÖĞRENMESI MODELI

- Artık tüm veri analizini ve hazırlığını tamamladığımıza göre, K-Means kümeleme modelini oluşturabiliriz.

Algoritmayı, PySpark'ın makine öğrenimi API'si kullanılarak oluşturacağız.

I) Kullanılacak K değerini bulma

Bir K-Means kümeleme modeli oluştururken, öncelikle algoritmanın döndürmesini istediğimiz küme veya grup sayısını belirlememiz gerekir. Örneğin üç kümeye karar verirsek, o zaman üç müşteri segmentimiz olacaktır.

K-Means'de kaç kümenin kullanılacağına karar vermek için kullanılan en popüler teknik "elbow yöntemi"'dur.

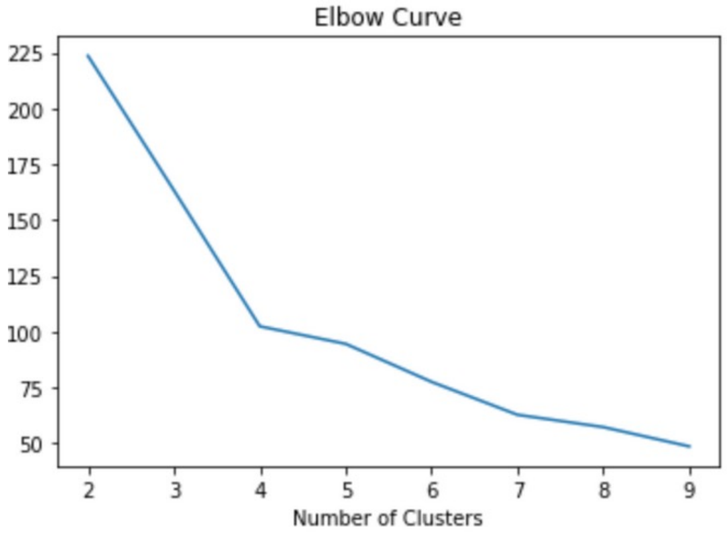
Bu, birçok kere K-Means algoritmasını çalıştırarak her küme için model sonuçlarını görselleştirir. Bu sayede optimum K sayısı ile dirsek yapılan noktadır.

2 ila 10 küme arasında bir K-Means kümeleme algoritmasını deniyelim:

K-MEANS

```
cost = np.zeros(10)

evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='standardized',metricName='silhouette',

for i in range(2,10):
    KMeans_algo=KMeans(featuresCol='standardized', k=i)
    KMeans_fit=KMeans_algo.fit(data_scale_output)
    output=KMeans_fit.transform(data_scale_output)
    cost[i] = KMeans_fit.summary.trainingCost
```

```
df_cost = pd.DataFrame(cost[2:])
df_cost.columns = ["cost"]
new_col = range(2,10)
df_cost.insert(0, 'cluster', new_col)
pl.plot(df_cost.cluster, df_cost.cost)
pl.xlabel('Number of Clusters')
pl.ylabel('Score')
pl.title('Elbow Curve')
pl.show()
```



- K değeri olarak 4 uygun gözüküyor.

K-MEANS

- Veri kümesindeki her müşteriye kümeler atamak için oluşturduğumuz modeli kullanalım:

```
KMeans_algo=KMeans(featuresCol='standardized', k=4)
KMeans_fit=KMeans_algo.fit(data_scale_output)
```

```
1 preds=KMeans_fit.transform(data_scale_output)
2
3 preds.show(5,0)
```

```
+-------+---------+-----------------+----------+--------------------------------+------------------------------
-------------------------+----------+

|recency|frequency|monetary_value    |CustomerID|features                        |standardized
|prediction|

+-------+---------+-----------------+----------+--------------------------------+------------------------------
-------------------------+----------+

|5580   |14       |226.14            |16250     |[5580.0,14.0,226.14]            |[0.6860448646904733,0.6848507976
304105,0.45968090513788235]|0         |

|2580   |1        |350.4             |15100     |[2580.0,1.0,350.4]              |[0.3172035395880683,0.0489179141
1645789,0.7122675738936676]|0         |

|30360  |14       |205.85999999999999|13065     |[30360.0,14.0,205.85999999999999]|[3.732674210036339,0.68485079763
04105,0.4184571996625297]  |2         |

|12660  |59       |390.78999999999985|12838     |[12660.0,59.0,390.78999999999985]|[1.556510391932149,2.88615693287
10157,0.7943694212383169]  |3         |

|18420  |5        |115.65            |15350     |[18420.0,5.0,115.65]            |[2.264685736128767,0.24458957058
228947,0.23508488847261033]|2         |

+-------+---------+-----------------+----------+--------------------------------+------------------------------
-------------------------+----------+

only showing top 5 rows
```

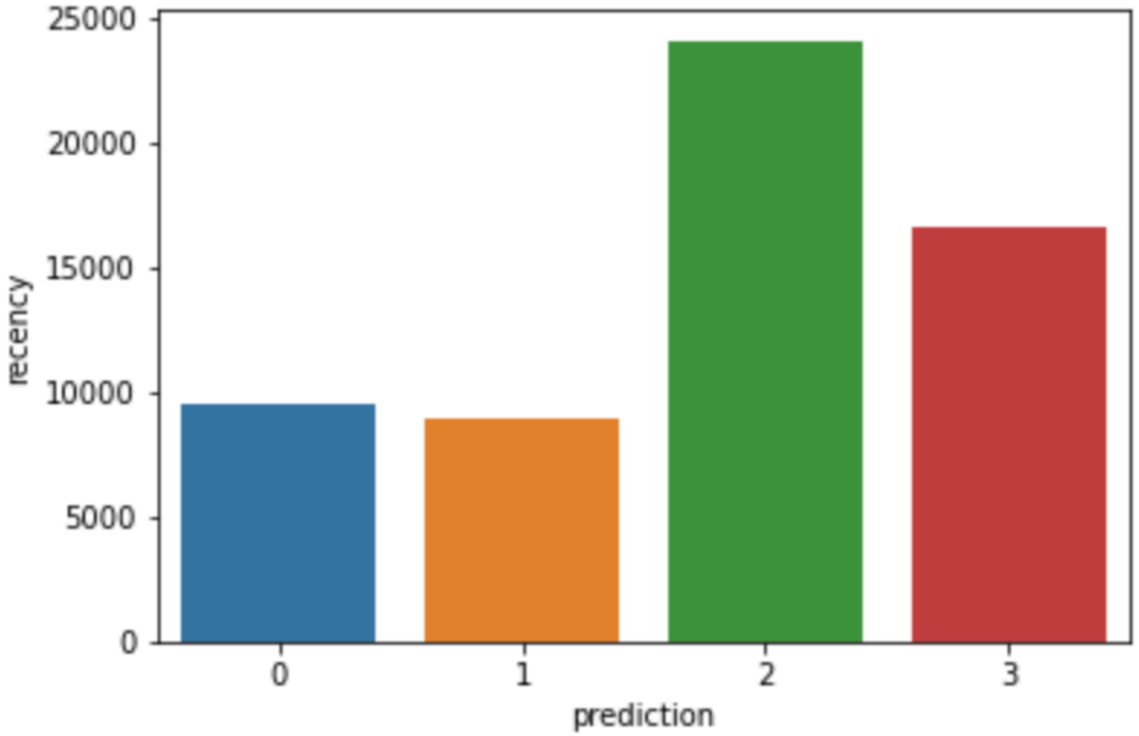
KÜME ANALIZI

```
df_viz = preds.select('recency','frequency','monetary_value','prediction')
df_viz = df_viz.toPandas()
avg_df = df_viz.groupby(['prediction'], as_index=False).mean()

list1 = ['recency','frequency','monetary_value']

for i in list1:
    sns.barplot(x='prediction',y=str(i),data=avg_df)
    plt.show()
```

- Son olarak, yeni oluşturduğumuz müşteri segmentlerini analiz edelim.

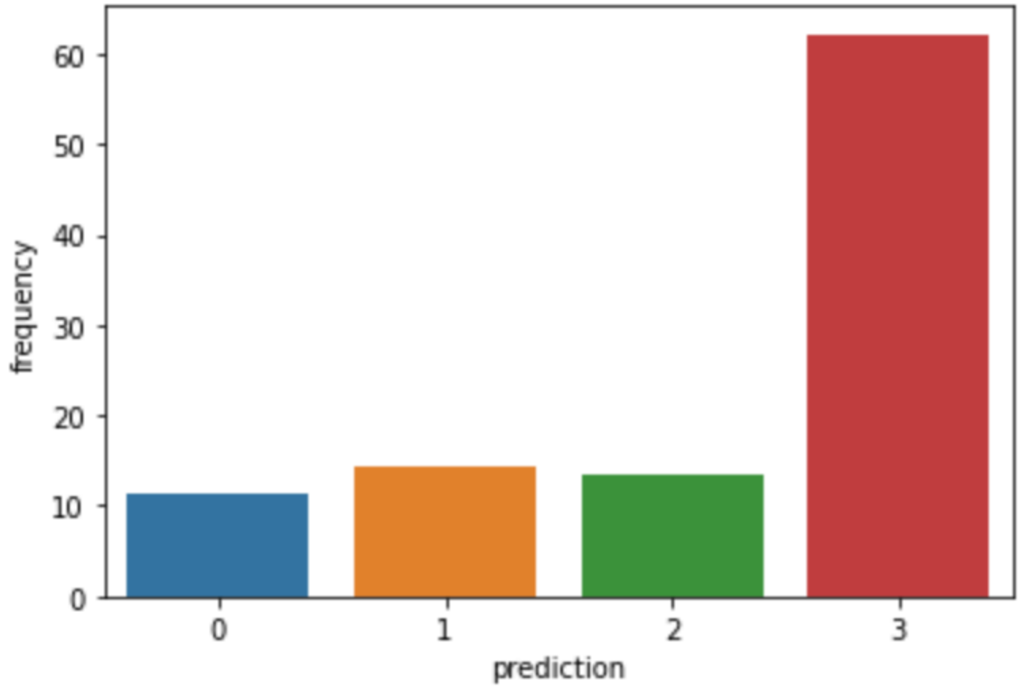
  Veri çerçevesindeki her bir müşteri kimliğinin recency, frequency ve monetary_value'ini görselleştirelim.

KÜME ANALIZI



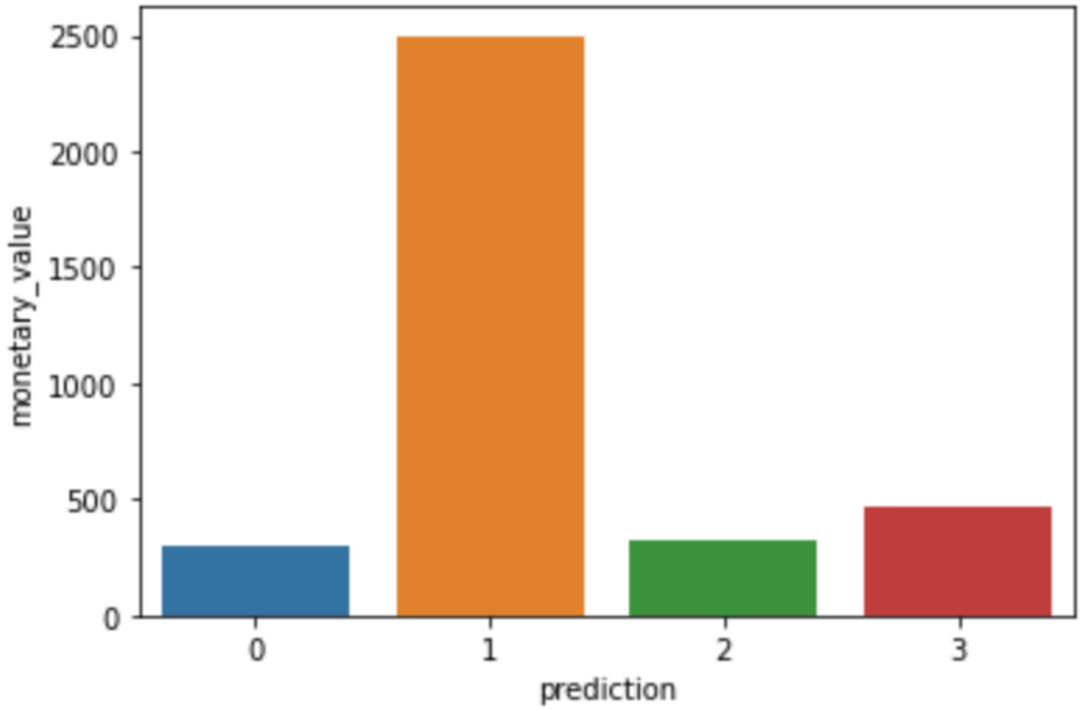
- Çıkan sonuçlarımızı tek tek inceleyelim ve son sayfada yorumlayalım.

KÜME ANALIZI



- Çıkan sonuçlarımızı tek tek inceleyelim ve son sayfada yorumlayalım.

KÜME ANALIZI



- Çıkan sonuçlarımızı tek tek inceleyelim ve son sayfada yorumlayalım.

KÜME ANALIZI

İşte her kümede müşteriler tarafından görüntülenen özelliklere genel bir bakış:

Küme 0: Bu segmentteki müşteriler düşük yenilik, sıklık ve parasal değer göstermekteler. Platformda nadiren alışveriş yapmışlar ve e-ticaret şirketiyle iş yapmayı bırakma olasılığı düşük potansiyelli müşterilerdir.

Küme 1: Bu kümedeki kullanıcılar yüksek yenilik gösteriyor ancak platformda fazla harcama yapmamışlardır. Ayrıca siteyi sık sık ziyaret de etmemektedirler. Bu, şirketle iş yapmaya yeni başlayan, daha yeni müşteriler olabileceklerini söyleyebiliriz.

Küme 2: Bu segmentteki müşteriler orta düzeyde yenilik ve sıklık göstermekte ve platformda çok para harcamaktalar. Bu, yüksek değerli ürünler satın alma veya toplu alım yapma eğiliminde olduklarını göstermektedir.

Küme 3: Son segment, yüksek yenilik gösteren ve platformda sık sık alışveriş yapan kullanıcıları içermektedir. Bununla birlikte, platformda fazla harcama yapmamakla birlikte, her satın alımda daha ucuz ürünler seçme eğiliminde oldukları anlamına da gelebilmektedir.

Teşekkürler!