Kevser Bahadır 20201101086

PROJENİN GENEL AMACI VE ÖZETİ

Bu projenin amacı, Julia yazılım dili ile istatistik bilgileri ve günümüzün en popüler makine öğrenmesi metodlarını farklı ve uygun veri setleri ile harmanlayarak çeşitli uygulamalar ile daha kolay anlaşılır hale getirmektir. Proje, istatistiksel yöntemleri kullanarak raporlama, doğrusal regresyon modellerinden başlayarak basit doğrusal regresyon, çoklu doğrusal regresyon, lojistik regresyon, karar ağaçları (CART), random forest, k-en yakın komşuluk algoritması ve kümeleme gibi temel makine öğrenmesi konseptlerini kapsamaktadır. Her bir modelin açıklaması ve örnek uygulamaları, kullanıcıların bu teknikleri daha iyi anlamasına ve uygulamasına yardımcı olacaktır.

İÇERİK

- İstatistiksel Raporlama
- Doğrusal Regresyon Modeli
- Basit Doğrusal Regresyon Modeli
- Çoklu Doğrusal Regresyon Modeli
- Lojistik Regresyon
- Karar Ağaçları(CART)
- Random Forest
- K-En Yakın Komşuluk Algoritması
- Kümeleme

Not: Her konu başlığının altında kullanılan data hakkında bilgilendirme ve kaynağı(giriş kısmı) belirtilmiştir. İstatistiksel Raporlama bölümü tek bir data üzerinden örnek verilerek yapılmıştır.

İSTATİSTİKSEL RAPORLAMA

Veri Adı = Bank Churners | Kredi Kartı Müşterileri

Bir banka yöneticisi, müşterilerinin kredi kartı hizmetlerini terk etmelerinden duyduğu rahatsızlığı ifade etmektedir.Yöneticinin istediği, müşteri kayıplarını önceden tahmin edebilmek, bankaya müşterilere daha iyi hizmet sunma ve olası ayrılık kararlarını engelleme fırsatı yakalamak istemektedir. Bu nedenle, müşterilerin ayrılma eğilimini önceden kestirebilen bir sistem kurmak, bankanın proaktif bir şekilde müşterilere yaklaşmasını sağlayarak daha etkili bir müşteri ilişkisi yönetimine imkân tanıyabilmek amaçtır. Özetle projenin amacı, müşteri kaybını önceden bilmek ve ona göre banka kendi politikasını önceden belirlemek istemesidir.

Veri Setinin Açıklaması

10127 gözlem sayısından oluşan veri, Kaggle sitesinden alınmıştır. Bizim dikkate alacağımız sütun içeriği aşağıdaki gibidir;

Veri Setinin Açıklaması

- **Clientnum(int):** Müşteri numarası
- **Customer Age(int):** Müşterinin yaşı
- **Gender(nominal):** Müşterinin cinsiyeti ("F" = kadın | "M" erkek)
- **Education Level(ordinal):** Müşterinin öğrenim durumu
  ("uneducated" = eğitimsiz | "unknown" = bilinmiyor | "high school" = lise | "college" = üniversite | "graduate" = mezun | "post graduate" = yüksek lisans | "doctorate" = doktora)
- **Marital Status(nominal):** Müşterinin medeni durumu ("Married" = Evli | "Single" = Bekar | "Divorced" = Boşanmış | "Unknown" = Bilinmiyor)
- **Card Category(ordinal):** Müşterinin kart türü ("blue" = mavi | "silver" = gümüş | "gold" = altın)
- **Months On Book(int):** Müşterinin banka ile ilişki süresi
- **Credit Limit(int):** Müşterinin kredi limiti
- **Total Revolving Bal(int):** Toplam döner sermaye
- **Total Trans AMT(int):** 12 ay içinde toplam transfer tutarı
- **Total Trans CT(int):** 12 ay içinde toplam işlem sayısı

Data Kaynağı: https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers?resource=download

In [3]:
```
using Pkg
Pkg.add("DataFrames")
Pkg.add("CSV")
Pkg.add("Statistics")
Pkg.add("MLDataUtils")
Pkg.add("Impute")
Pkg.add("Plots")
```

```
    Updating registry at `C:\Users\Kev\.julia\registries\General`
    Updating git-repo `https://github.com/JuliaRegistries/General.git`
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 2 seconds (141 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 1 seconds (141 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 1 seconds (141 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
```

```
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
 0 dependencies successfully precompiled in 1 seconds (141 already precompiled, 6 skipped during auto due to previous errors)
 1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
 0 dependencies successfully precompiled in 1 seconds (141 already precompiled, 6 skipped during auto due to previous errors)
 1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
 0 dependencies successfully precompiled in 1 seconds (141 already precompiled, 6 skipped during auto due to previous errors)
 1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
```

In [39]:
```
using CSV
using DataFrames

# Doğru dosya yolunu belirtin
csv_dosya_yolu = "C:/Users/Kev/Desktop/BankChurners.csv"

# CSV dosyasını DataFrame'e yükle
df = DataFrame(CSV.File(csv_dosya_yolu))
```

Out[39]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	String	String	String	String	String	String
1	45,00	0,00	High School	Married	Blue	39,00
2	49,00	1,00	Graduate	Single	Blue	44,00
3	51,00	0,00	Graduate	Married	Blue	36,00
4	40,00	1,00	High School	Unknown	Blue	34,00
5	40,00	0,00	Uneducated	Married	Blue	21,00
6	44,00	0,00	Graduate	Married	Blue	36,00
7	51,00	0,00	Unknown	Married	Gold	46,00
8	32,00	0,00	High School	Unknown	Silver	27,00
9	37,00	0,00	Uneducated	Single	Blue	36,00
10	48,00	0,00	Graduate	Single	Blue	36,00
11	42,00	0,00	Uneducated	Unknown	Blue	31,00
12	65,00	0,00	Unknown	Married	Blue	54,00
13	56,00	0,00	College	Single	Blue	36,00
14	35,00	0,00	Graduate	Unknown	Blue	30,00
15	57,00	1,00	Graduate	Married	Blue	48,00
16	44,00	0,00	Unknown	Unknown	Blue	37,00
17	48,00	0,00	Post-Graduate	Single	Blue	36,00
18	41,00	0,00	Unknown	Married	Blue	34,00
19	61,00	0,00	High School	Married	Blue	56,00
20	45,00	1,00	Graduate	Married	Blue	37,00
21	47,00	0,00	Doctorate	Divorced	Blue	42,00
22	62,00	1,00	Graduate	Married	Blue	49,00
23	41,00	0,00	High School	Married	Blue	33,00
24	47,00	1,00	Unknown	Single	Blue	36,00
25	54,00	0,00	Unknown	Married	Blue	42,00
26	41,00	1,00	Graduate	Single	Blue	28,00
27	59,00	0,00	High School	Unknown	Blue	46,00
28	63,00	0,00	Unknown	Married	Blue	56,00
29	44,00	1,00	Uneducated	Single	Blue	34,00
30	47,00	0,00	High School	Married	Blue	42,00
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [155]:
```
size(df)
```

Out[155]: (10127, 10)

In [158]:
```
names(df)
```

Out[158]:
```
10-element Vector{String}:
 "Customer_Age"
 "Gender"
 "Education_Level"
 "Marital_Status"
 "Card_Category"
 "Months_on_book"
 "Credit_Limit"
 "Total_Revolving_Bal"
 "Total_Trans_Amt"
 "Total_Trans_Ct"
```

In [22]:
```
show(describe(df,:all),allrows=true, allcols=true)
```

```
10×13 DataFrame
 Row │ variable            mean     std      min       q25      median
     │ Symbol              Union…   Union…   Any       Union…   Union…
─────┼─────────────────────────────────────────────────────────────────
   1 │ Customer_Age        46.326   8.01681  26        41.0     46.0
   2 │ Gender                                F
   3 │ Education_Level                       College
   4 │ Marital_Status                        Divorced
   5 │ Card_Category                         Blue
   6 │ Months_on_book      35.9284  7.98642  13        31.0     36.0
   7 │ Credit_Limit        8631.95  9088.78  1438.3    2555.0   4549.0
   8 │ Total_Revolving_Bal 1162.81  814.987  0         359.0    1276.0
   9 │ Total_Trans_Amt     4404.09  3397.13  510       2155.5   3899.0
  10 │ Total_Trans_Ct      64.8587  23.4726  10        45.0     67.0

 Row │ q75      max      nunique  nmissing  first        last
     │ Union…   Any      Union…   Nothing   Any          Any
─────┼──────────────────────────────────────────────────────────
   1 │ 52.0     73                          45           43
   2 │          M        2                  M            F
   3 │          Unknown  7                  High School  Graduate
   4 │          Unknown  4                  Married      Married
   5 │          Silver   4                  Blue         Silver
   6 │ 40.0     56                          39           25
   7 │ 11067.5  34516.0                     12691.0      10388.0
   8 │ 1784.0   2517                        777          1961
   9 │ 4741.0   18484                       1144         10294
  10 │ 81.0     139                         42           61

 Row │ eltype
     │ DataType
─────┼──────────
   1 │ Int64
   2 │ String1
   3 │ String15
   4 │ String15
   5 │ String15
   6 │ Int64
   7 │ Float64
   8 │ Int64
   9 │ Int64
  10 │ Int64
```

Tekrar Eden Gözlemleri Silmek (Remove Duplicates)

In [160]:
```
using DataFrames
df=unique!(df)
```

Out[160]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	String1	String15	String15	String15	Int64
1	45	M	High School	Married	Blue	39
2	49	F	Graduate	Single	Blue	44
3	51	M	Graduate	Married	Blue	36
4	40	F	High School	Unknown	Blue	34
5	40	M	Uneducated	Married	Blue	21
6	44	M	Graduate	Married	Blue	36
7	51	M	Unknown	Married	Gold	46
8	32	M	High School	Unknown	Silver	27
9	37	M	Uneducated	Single	Blue	36
10	48	M	Graduate	Single	Blue	36
11	42	M	Uneducated	Unknown	Blue	31
12	65	M	Unknown	Married	Blue	54
13	56	M	College	Single	Blue	36
14	35	M	Graduate	Unknown	Blue	30
15	57	F	Graduate	Married	Blue	48
16	44	M	Unknown	Unknown	Blue	37
17	48	M	Post-Graduate	Single	Blue	36
18	41	M	Unknown	Married	Blue	34
19	61	M	High School	Married	Blue	56
20	45	F	Graduate	Married	Blue	37
21	47	M	Doctorate	Divorced	Blue	42
22	62	F	Graduate	Married	Blue	49
23	41	M	High School	Married	Blue	33
24	47	F	Unknown	Single	Blue	36
25	54	M	Unknown	Married	Blue	42
26	41	F	Graduate	Single	Blue	28
27	59	M	High School	Unknown	Blue	46
28	63	M	Unknown	Married	Blue	56
29	44	F	Uneducated	Single	Blue	34
30	47	M	High School	Married	Blue	42
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [24]:
```
describe(df,:nmissing)
```
Out[24]: 10 rows × 2 columns

	variable	nmissing
	Symbol	Nothing
1	Customer_Age	
2	Gender	
3	Education_Level	
4	Marital_Status	
5	Card_Category	
6	Months_on_book	
7	Credit_Limit	
8	Total_Revolving_Bal	
9	Total_Trans_Amt	
10	Total_Trans_Ct	

In [161]:
```
filter(row-> any(ismissing, row), df)
```
Out[161]: 0 rows × 10 columns (omitted printing of 4 columns)

Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
Int64	String1	String15	String15	String15	Int64

In [163]:
```
ismissing.(df)
```
Out[163]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Bool	Bool	Bool	Bool	Bool	Bool
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0

Kayıp Gözlemleri Impute Paketi İle İnceleme

In [27]:
```
using DataFrames, Impute
df2=Impute.interp(df)
```
Out[27]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	String1	String15	String15	String15	Int64
1	45	M	High School	Married	Blue	39
2	49	F	Graduate	Single	Blue	44
3	51	M	Graduate	Married	Blue	36
4	40	F	High School	Unknown	Blue	34
5	40	M	Uneducated	Married	Blue	21
6	44	M	Graduate	Married	Blue	36
7	51	M	Unknown	Married	Gold	46
8	32	M	High School	Unknown	Silver	27
9	37	M	Uneducated	Single	Blue	36
10	48	M	Graduate	Single	Blue	36

In [164]:
```
describe(df2,:nmissing)
```
Out[164]: 10 rows × 2 columns

	variable	nmissing
	Symbol	Nothing
1	Customer_Age	
2	Gender	
3	Education_Level	
4	Marital_Status	
5	Card_Category	
6	Months_on_book	
7	Credit_Limit	
8	Total_Revolving_Bal	
9	Total_Trans_Amt	
10	Total_Trans_Ct	

Grafik Çizdirme

In [15]:
```
Pkg.add("StatsPlots")
using StatsPlots
```
```
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
```

In [31]:
```
plot(df[!,8], seriestype=:scatter, title="Credit_Limit")
```
Out[31]:

In [166]:
```
# Histogram çizimi
histogram(df.Customer_Age, xlabel="Customer_Age", ylabel="Frequency", legend=false)
```
Out[166]:

In [21]:
```
# Kutu grafiği çizimi
boxplot(df.Customer_Age, xlabel="Customer_Age", ylabel="Age", legend=false)
```
Out[21]:

Aykırı değer gözlemlenmektedir. Ortalama yaş 40-50 yaş arasında değişmektedir.

In [167]:
```
# Çizgi grafiği
plot(df.Customer_Age, xlabel="Index", ylabel="Age", label="Customer Age", seriestype=:line)
```
Out[167]:

In [168]:
```
plot(df[!,9], seriestype=:scatter, title="Total_Revolving_Bal")
```
Out[168]:

İstatistiksel Analiz

In [170]:
```
using Statistics
df2=df[:,mean.(ismissing,eachcol(df)).<0.00001]
```
Out[170]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	String1	String15	String15	String15	Int64
1	45	M	High School	Married	Blue	39
2	49	F	Graduate	Single	Blue	44
3	51	M	Graduate	Married	Blue	36
4	40	F	High School	Unknown	Blue	34
5	40	M	Uneducated	Married	Blue	21
6	44	M	Graduate	Married	Blue	36
7	51	M	Unknown	Married	Gold	46
8	32	M	High School	Unknown	Silver	27
9	37	M	Uneducated	Single	Blue	36
10	48	M	Graduate	Single	Blue	36

Ortalama, minimum, maximum , medyan, kayıp veri ve data tipleri yukarıdaki gibidir.

In [171]:
```
show(describe(df2), allrows=true, allcols=true)
```
```
10×8 DataFrame
 Row │ variable            mean     min       median  max       nunique
     │ Symbol              Union…   Any       Union…  Any       Union…
─────┼─────────────────────────────────────────────────────────────────
   1 │ Customer_Age        46.326   26        46.0    73
   2 │ Gender                       F                 M         2
   3 │ Education_Level              College           Unknown   7
   4 │ Marital_Status               Divorced          Unknown   4
   5 │ Card_Category                Blue              Silver    4
   6 │ Months_on_book      35.9284  13        36.0    56
   7 │ Credit_Limit        8631.95  1438.3    4549.0  34516.0
   8 │ Total_Revolving_Bal 1162.81  0         1276.0  2517
   9 │ Total_Trans_Amt     4404.09  510       3899.0  18484
  10 │ Total_Trans_Ct      64.8587  10        67.0    139

 Row │ nmissing  eltype
     │ Nothing   DataType
─────┼────────────────────
   1 │           Int64
   2 │           String1
   3 │           String15
   4 │           String15
   5 │           String15
   6 │           Int64
   7 │           Float64
   8 │           Int64
   9 │           Int64
  10 │           Int64
```

In [172]:
```
(ismissing,eachcol(df))
```

Out[172]:
```
(ismissing, 10127×10 DataFrameColumns. Omitted printing of 6 columns
│ Row   │ Customer_Age │ Gender  │ Education_Level │ Marital_Status │
│       │ Int64        │ String1 │ String15        │ String15       │
├───────┼──────────────┼─────────┼─────────────────┼────────────────┤
│ 1     │ 45           │ M       │ High School     │ Married        │
│ 2     │ 49           │ F       │ Graduate        │ Single         │
│ 3     │ 51           │ M       │ Graduate        │ Married        │
│ 4     │ 40           │ F       │ High School     │ Unknown        │
│ 5     │ 40           │ M       │ Uneducated      │ Married        │
│ 6     │ 44           │ M       │ Graduate        │ Married        │
│ 7     │ 51           │ M       │ Unknown         │ Married        │
│ 8     │ 32           │ M       │ High School     │ Unknown        │
│ 9     │ 37           │ M       │ Uneducated      │ Single         │
│ 10    │ 48           │ M       │ Graduate        │ Single         │
⋮
│ 10117 │ 46           │ M       │ College         │ Single         │
│ 10118 │ 57           │ M       │ Graduate        │ Married        │
│ 10119 │ 50           │ M       │ Unknown         │ Unknown        │
│ 10120 │ 55           │ F       │ Uneducated      │ Single         │
│ 10121 │ 54           │ M       │ High School     │ Single         │
│ 10122 │ 56           │ F       │ Graduate        │ Single         │
│ 10123 │ 50           │ M       │ Graduate        │ Single         │
│ 10124 │ 41           │ M       │ Unknown         │ Divorced       │
│ 10125 │ 44           │ F       │ High School     │ Married        │
│ 10126 │ 30           │ M       │ Graduate        │ Unknown        │
│ 10127 │ 43           │ F       │ Graduate        │ Married        │)
```

In [173]:
```
using Statistics
df=df[:,mean.(ismissing,eachcol(df)).<0.1]
```

Out[173]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	String1	String15	String15	String15	Int64
1	45	M	High School	Married	Blue	39
2	49	F	Graduate	Single	Blue	44
3	51	M	Graduate	Married	Blue	36
4	40	F	High School	Unknown	Blue	34
5	40	M	Uneducated	Married	Blue	21
6	44	M	Graduate	Married	Blue	36
7	51	M	Unknown	Married	Gold	46
8	32	M	High School	Unknown	Silver	27
9	37	M	Uneducated	Single	Blue	36
10	48	M	Graduate	Single	Blue	36

In [16]:
```
using Pkg
Pkg.add("Impute")
```

```
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
```

In [174]:
```
df
```

Out[174]: 10,127 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	String1	String15	String15	String15	Int64
1	45	M	High School	Married	Blue	39
2	49	F	Graduate	Single	Blue	44
3	51	M	Graduate	Married	Blue	36
4	40	F	High School	Unknown	Blue	34
5	40	M	Uneducated	Married	Blue	21
6	44	M	Graduate	Married	Blue	36
7	51	M	Unknown	Married	Gold	46
8	32	M	High School	Unknown	Silver	27
9	37	M	Uneducated	Single	Blue	36
10	48	M	Graduate	Single	Blue	36

In [175]:
```
filter(row-> any(ismissing, row), df)
```

Out[175]: 0 rows × 10 columns (omitted printing of 4 columns)

Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
Int64	String1	String15	String15	String15	Int64

Çeyrekler Arası Aralığı Bulma

Veri setindeki çeyrekler arası aralığı (IQR) hesaplanmaktadır.

In [176]:
```
function calculate_iqr(dataframe, column)
    q1 = quantile(dataframe[!, column], 0.25)
    q3 = quantile(dataframe[!, column], 0.75)
    iqr_value = q3 - q1
    return iqr_value
end
```

Out[176]:
```
calculate_iqr (generic function with 1 method)
```

In [177]:
```
# Customer_Age sütunundaki çeyrekler arası aralığı hesapla
customer_age_iqr = calculate_iqr(df, :Customer_Age)

println("Customer_Age Çeyrekler Arası Aralık: $customer_age_iqr")
```

```
Customer_Age Çeyrekler Arası Aralık: 11.0
```

In [178]:
```
# Customer_Age sütunundaki mod'u bulma
```

In [17]:
```
using Pkg
Pkg.add("StatsBase")
```

```
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
```

Varyans bulma

In [45]:
```
customer_age_variance = var(df[!, :Customer_Age])

println("Customer_Age Sütunundaki Varyans: $customer_age_variance")
```

```
Customer_Age Sütunundaki Varyans: 64.26930723247507
```

In [46]:
```
Total_Revolving_Bal_variance = var(df[!, :Total_Revolving_Bal])
println("Total_Revolving_BalSütunundaki Varyans: $Total_Revolving_Bal_variance")
```

```
Total_Revolving_BalSütunundaki Varyans: 664204.3565946727
```

En çok tekrar eden değeri bulma

In [18]:
```
using Pkg
Pkg.add("StatsBase")
using StatsBase
```

```
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
```

In [48]:
```
using Statistics
gender_mode = StatsBase.mode(df[!, :Gender])

println("Gender Sütunundaki Mod: $gender_mode")
```

```
Gender Sütunundaki Mod: F
```

In [49]:
```
using Statistics
Education_Level_mode = StatsBase.mode(df[!, :Education_Level])

println("Education_Level Sütunundaki Mod: $Education_Level_mode")
```

```
Education_Level Sütunundaki Mod: Graduate
```

In [50]:
```
using Statistics
Marital_Status_mode = StatsBase.mode(df[!, :Marital_Status])

println("Marital_Status Sütunundaki Mod: $Marital_Status_mode")
```

```
Marital_Status Sütunundaki Mod: Married
```

Değişkenler Arasında Korelasyon Bulma

Customer_Age ve Credit_Limit sütunları arasındaki korelasyon katsayısını bulalım.

In [51]:
```
correlation_coefficient = cor(df[!, :Customer_Age], df[!, :Credit_Limit])
println("Customer_Age ve Credit_Limit Korelasyon Katsayısı: $correlation_coefficient")
```

```
Customer_Age ve Credit_Limit Korelasyon Katsayısı: 0.0024762273596652434
```

Customer_Age ve Credit_Limit Korelasyon Katsayısı: 0.0024762273596652434'dır. Aralarında pozitif zayıf bir bağ vardır.

Customer_Age ve Total_Trans_Amt sütunları arasındaki korelasyon katsayısını bulalım.

In [53]:
```
correlation_coefficient = cor(df[!, :Customer_Age], df[!, :Total_Trans_Amt])
println("Customer_Age ve Total_Trans_Amt Korelasyon Katsayısı: $correlation_coefficient")
```

```
Customer_Age ve Total_Trans_Amt Korelasyon Katsayısı: -0.046446490854686884
```

Customer_Age ve Total_Trans_Amt Korelasyon Katsayısı: -0.046446490854686884 Aralarında negatif zayıf bir bağ vardır.

DOĞRUSAL REGRESYON MODELİ

Doğrusal regresyon, bağımsız değişkenlerin ve bağımlı değişkenin arasındaki ilişkiyi modellemek ve bu ilişkiyi kullanarak bağımlı değişkenin değerini tahmin etmek için kullanılır.

Basit Doğrusal Regresyon Modeli

Basit doğrusal regresyon modeli, bir bağımlı değişkenin yalnızca bir bağımsız değişken tarafından açıklanmaya çalışıldığı bir regresyon modelidir.

DATA

Bank Churners(Kredi Kartı Müşterileri) kullanmaya devam edebiliriz. Datatype olarak da uygundur hem kategorik hem numerik değerleri içerir. Total_Trans_Amt (toplam transfer tutarı) bağımlı değişken, Total_Trans_Ct (müşterinin işlem sayısı) ise bağımsız değişken olarak düşünerek işlemlerimizi sürdüreceğiz. Örneğin; bir müşterinin yaptığı toplam işlem sayısının, bu işlemlerin toplam transfer tutarı üzerindeki etkisini anlamak için bir regresyon analizi yapabilirsiniz. Bu tür bir analiz, işlemlerle transfer tutarı arasındaki ilişkiyi anlamanıza ve tahmin etmenize yardımcı olabilir.

In [19]:
```
using Pkg

Pkg.add("DataFrames")
Pkg.add("CSV")
Pkg.add("Plots")
Pkg.add("Lathe")
Pkg.add("GLM")
Pkg.add("StatsPlots")
Pkg.add("MLBase")
Pkg.add("Statistics")



using DataFrames
using CSV
using Plots
using Lathe
using GLM
using Statistics
using StatsPlots
using MLBase
```

```
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
   Resolving package versions...
```

```
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  Resolving package versions...
```

In [55]:
```
describe(df)
```

Out[55]: 10 rows × 8 columns

	variable	mean	min	median	max	nunique	nmissing	eltype
	Symbol	Union...	Any	Union...	Any	Union...	Nothing	DataType
1	Customer_Age	46.326	26	46.0	73			Int64
2	Gender		F		M	2		String1
3	Education_Level		College		Unknown	7		String15
4	Marital_Status		Divorced		Unknown	4		String15
5	Card_Category		Blue		Silver	4		String15
6	Months_on_book	35.9284	13	36.0	56			Int64
7	Credit_Limit	8631.95	1438.3	4549.0	34516.0			Float64
8	Total_Revolving_Bal	1162.81	0	1276.0	2517			Int64
9	Total_Trans_Amt	4404.09	510	3899.0	18484			Int64
10	Total_Trans_Ct	64.8587	10	67.0	139			Int64

Doğrusal İlişkinin Varlığı

In [59]:
```
scatter(df.Total_Trans_Amt, df.Total_Trans_Ct, xlabel="12 ay içinde toplam transfer tutarı", ylabel="Müşterinin işlem sayısı ",le
```

Out[59]:

In [61]:
```
using Statistics
cor(df.Total_Trans_Amt,df.Total_Trans_Ct)
```

Out[61]: 0.8071920346514343

Dağılım Analizi

In [62]:
```
density(df.Total_Trans_Ct, title="Yoğunluk Grafiği", ylabel="toplam transfer tutarı", xlabel="Müşterinin işlem sayısı",legend=fal
```

Out[62]:

Veriyi Bölme (Train-Test)

In [63]:
```
using Lathe.preprocess: TrainTestSplit
train,test=TrainTestSplit(df,.75)
```

Out[63]:
```
(7651×10 DataFrame. Omitted printing of 6 columns
 │ Row  │ Customer_Age │ Gender  │ Education_Level │ Marital_Status │
 │      │ Int64        │ String1 │ String15        │ String15       │
 ├──────┼──────────────┼─────────┼─────────────────┼────────────────┤
 │ 1    │ 45           │ M       │ High School     │ Married        │
 │ 2    │ 49           │ F       │ Graduate        │ Single         │
 │ 3    │ 51           │ M       │ Graduate        │ Married        │
 │ 4    │ 40           │ F       │ High School     │ Unknown        │
 │ 5    │ 40           │ M       │ Uneducated      │ Married        │
 │ 6    │ 51           │ M       │ Unknown         │ Married        │
 │ 7    │ 37           │ M       │ Uneducated      │ Single         │
 │ 8    │ 48           │ M       │ Graduate        │ Single         │
 │ 9    │ 42           │ M       │ Uneducated      │ Unknown        │
 │ 10   │ 65           │ M       │ Unknown         │ Married        │
 ⋮
 │ 7641 │ 29           │ M       │ Graduate        │ Married        │
 │ 7642 │ 38           │ M       │ Uneducated      │ Single         │
 │ 7643 │ 46           │ M       │ College         │ Single         │
 │ 7644 │ 57           │ M       │ Graduate        │ Married        │
```

Total_Trans_Amt (toplam transfer tutarı) bağımlı değişken, Total_Trans_Ct (müşterinin işlem sayısı) ise bağımsız değişken olarak düşünelim, demiştik.

In [64]:
```
using GLM
fm=@formula(Total_Trans_Amt~Total_Trans_Ct) # İlk argüman bağımlı değişken, ikincisi bağımsız değişken.
linreg=lm(fm,train)
```

Out[64]:
```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgebra.CholeskyPivo
ted{Float64, Matrix{Float64}}}}, Matrix{Float64}}

Total_Trans_Amt ~ 1 + Total_Trans_Ct

Coefficients:
──────────────────────────────────────────────────────────────────────────
                    Coef.  Std. Error       t  Pr(>|t|)  Lower 95%  Upper 95%
──────────────────────────────────────────────────────────────────────────
(Intercept)     -3210.91    67.162    -47.81    <1e-99  -3342.56   -3079.25
Total_Trans_Ct    117.574    0.973705  120.75    <1e-99    115.665    119.483
──────────────────────────────────────────────────────────────────────────
```

R2'yi Hesaplama

In [66]:
```
r2(linreg)
```

Out[66]: 0.6559049639579237

Tahmin Etme

In [67]:
```
?predict();
```

In [68]:
```
test_pred=predict(linreg,test)
```

Out[68]:
```
2476-element Vector{Union{Missing, Float64}}:
  -389.1341321137338
  1021.7530382432515
   198.7355222016763
  -741.8559247029802
 -1329.7255790183906
  -506.7080629768161
  1256.9008999694151
  1844.7705542848253
    81.16159133859446
 -1329.7255790183906
   551.4573147909227
  -977.0037864291444
 -1447.2995098814727
     ⋮
 11368.258954194473
 12191.276470236047
  9134.354267795916
 10662.815369015982
```

In [78]:
```
train_pred=predict(linreg,train)
```

Out[78]:
```
7651-element Vector{Union{Missing, Float64}}:
  1727.1966234217434
   669.031245654005
  -859.4298555660621
  -859.4298555660621
    81.16159133859446
   433.88338392784044
  -389.1341321137338
   551.4573147909227
  1727.1966234217434
  -153.98627038756968
 -1212.1516481553085
   669.031245654005
   -36.412339524487834
     ⋮
  6782.875650534273
  8076.1888900281765
  9604.649991248243
  9957.37178383749
```

In [70]:
```
perf_test=DataFrame(y_original=test[!,:Total_Trans_Amt],y_pred=test_pred)
```

Out[70]: 2,476 rows × 2 columns

	y_original	y_pred
	Int64	Float64?
1	1088	-389.134
2	1538	1021.75
3	1570	198.736
4	1207	-741.856
5	692	-1329.73
6	1197	-506.708
7	1045	1256.9
8	1407	1844.77
9	1464	81.1616
10	704	-1329.73

In [71]:
```
perf_test.hata=perf_test[!,:y_original]-perf_test[!,:y_pred]
```

Out[71]:
```
2476-element Vector{Float64}:
 1477.1341321137338
  516.2469617567485
 1371.2644777983237
 1948.8559247029802
 2021.7255790183906
 1703.708062976816
 -211.90089996941515
 -437.77055428482527
 1382.8384086614055
 2033.7255790183906
 1204.5426852090773
 1682.0037864291444
 2049.2995098814727
    ⋮
 4950.741045805527
 3085.7235297639527
 4805.645732204084
 5681.184630984018
```

```
In [72]: perf_test.hata_kare=perf_test.hata.*perf_test.hata
```

```
Out[72]: 2476-element Vector{Float64}:
      2.1819252442553937e6
 266510.92552307376
      1.8803662680715094e6
      3.7980394152499083e6
      4.0873743168572467e6
      2.902621163852215e6
  44901.99140784809
 191643.05819884315
      1.9122420644692085e6
      4.136039730753688e6
      1.4509230804906941e6
      2.829136737561979e6
      4.1996284812004445e6
      ⋮
      2.450983690262361e7
      9.521689702138908e6
      2.3094230903451327e7
      3.227585881132902e7
```

```
In [73]: perf_test
```

Out[73]: 2,476 rows × 4 columns

	y_original	y_pred	hata	hata_kare
	Int64	Float64?	Float64	Float64
1	1088	-389.134	1477.13	2.18193e6
2	1538	1021.75	516.247	2.66511e5
3	1570	198.736	1371.26	1.88037e6
4	1207	-741.856	1948.86	3.79804e6
5	692	-1329.73	2021.73	4.08737e6
6	1197	-506.708	1703.71	2.90262e6
7	1045	1256.9	-211.901	44902.0
8	1407	1844.77	-437.771	1.91643e5
9	1464	81.1616	1382.84	1.91224e6
10	704	-1329.73	2033.73	4.13604e6

```
In [79]: perf_train=DataFrame(y_original=train[!,:Total_Trans_Amt],y_pred=train_pred)
```

Out[79]: 7,651 rows × 2 columns

	y_original	y_pred
	Int64	Float64?
1	1144	1727.2
2	1291	669.031
3	1887	-859.43
4	1171	-859.43
5	816	81.1616
6	1330	433.883
7	1350	-389.134
8	1441	551.457
9	1201	1727.2
10	1314	-153.986

```
In [80]: perf_train.hata=perf_train[!,:y_original]-perf_train[!,:y_pred]
```

```
Out[80]: 7651-element Vector{Float64}:
 -583.1966234217434
  621.968754345995
 2746.429855566062
 2030.429855566062
  734.8384086614055
  896.1166160721596
 1739.1341321137338
  889.5426852090773
 -526.1966234217434
 1467.9862703875697
 2751.1516481553085
  641.968754345995
 1384.4123395244878
    ⋮
 3436.1243494657274
 6646.8111099718235
 7023.350008751757
 5396.6282161625095
```

In [81]:
```
perf_train.hata_kare=perf_train.hata.*perf_train.hata
```

Out[81]:
```
7651-element Vector{Float64}:
 340118.30157052283
 386845.1313827086
      7.542876951544621e6
      4.1226453983740197e6
 539987.4868440268
 803024.9896006183
      3.02458752948299e6
 791286.1888089755
 276882.8865004441
      2.154983690046407e6
      7.56883539114767e6
 412123.8815565484
      1.9165975258276658e6
      ⋮
      1.1806950544991268e7
      4.4180097931644864e7
      4.93274453454333e7
      2.912359610348135e7
```

In [82]:
```
perf_train
```

Out[82]: 7,651 rows × 4 columns

	y_original	y_pred	hata	hata_kare
	Int64	Float64?	Float64	Float64
1	1144	1727.2	-583.197	3.40118e5
2	1291	669.031	621.969	3.86845e5
3	1887	-859.43	2746.43	7.54288e6
4	1171	-859.43	2030.43	4.12265e6
5	816	81.1616	734.838	5.39987e5
6	1330	433.883	896.117	803025.0
7	1350	-389.134	1739.13	3.02459e6
8	1441	551.457	889.543	7.91286e5
9	1201	1727.2	-526.197	2.76883e5
10	1314	-153.986	1467.99	2.15498e6

Performans Ölçümü

MAE (Mean Absolute Error-Ortalama Mutlak Hata)

MAPE(Mean Absolute Percentage Error - Ortalama Mutlak Yüzde Hata)

In [83]:
```
function mape(perf_df)
    mape=mean(abs.(perf_df.hata./perf_df.y_original))
    return mape
end
```

Out[83]:
```
mape (generic function with 1 method)
```

In [84]:
```
mape(perf_test)
```

Out[84]:
```
0.4068214071100352
```

RMSE (Root Mean Square Error(Deviation) - Kök Ortalama Kare Hatası)

In [86]:
```
function rmse(perf_df)
    rmse=sqrt(mean(perf_df.hata.*perf_df.hata))
    return rmse
end
```

Out[86]:
```
rmse (generic function with 1 method)
```

MSE(Mean Square Error-Hata Kareleri Ortalaması)

Hata Analizi

In [90]:
```
#Test hatalarının karşılaştırılması
println("Ortalama Mutlak Test Hatası: ",mean(abs.(perf_test.hata))) #MAE
println("Ortalama Mutlak Yüzde Test Hatası: ",mape(perf_test))      #MAPE
println("Kök Ortalama Kare Test Hatası: ", rmse(perf_test))         #RMSD
println("Hata Kareleri Ortalaması Test Hatası: ", mean(perf_test.hata_kare))    #MSE
```

```
Ortalama Mutlak Test Hatası: 1480.7757594744112
Ortalama Mutlak Yüzde Test Hatası: 0.4068214071100352
Kök Ortalama Kare Test Hatası: 2003.3479204355567
Hata Kareleri Ortalaması Test Hatası: 4.01340289031347e6
```

Hataların Dağılımı

In [91]:
```
#Test Hata dağılımı
histogram(perf_test.hata,title="Test Hata Analizi",ylabel="Frekans",xlabel="Hata",legend=true)
```

Out[91]:

In [92]:
```
#Train Hataları Karşılaştırması
println("Ortalama Mutlak Train Hatası: ",mean(abs.(perf_train.hata)))
println("Ortalama Mutlak Yüzde Train Hatası: ",mape(perf_train))
println("Kök Ortalama Kare Train Hatası: ", rmse(perf_train))
println("Hata Kareleri Ortalaması Train Hatası: ", mean(perf_train.hata_kare))
```

```
Ortalama Mutlak Train Hatası: 1481.400473977008
Ortalama Mutlak Yüzde Train Hatası: 0.40606830591318527
Kök Ortalama Kare Train Hatası: 2005.9244050795019
Hata Kareleri Ortalaması Train Hatası: 4.0237327188935536e6
```

In [93]:
```
#Train Hata Dağılımı
histogram(perf_train.hata,title="Train Hata Analizi",ylabel="Frekans",xlabel="Hata",legend=false)
```

Out[93]:

In [94]:
```
train
```

Out[94]: 7,651 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	String1	String15	String15	String15	Int64
1	45	M	High School	Married	Blue	39
2	49	F	Graduate	Single	Blue	44
3	51	M	Graduate	Married	Blue	36
4	40	F	High School	Unknown	Blue	34
5	40	M	Uneducated	Married	Blue	21
6	51	M	Unknown	Married	Gold	46
7	37	M	Uneducated	Single	Blue	36
8	48	M	Graduate	Single	Blue	36
9	42	M	Uneducated	Unknown	Blue	31
10	65	M	Unknown	Married	Blue	54

Cross Validation (Çapraz Doğrulama)

In [107]:
```
function cross_validation(df,k,fm)
    a=collect(Kfold(size(df)[1],k))
    hata=[]
    for i in 1:k
        row=a[i]
        temp_train=df[row,:]
        temp_test=df[setdiff(1:end,row),:]
        linreg=lm(fm,temp_train)
        perf_test=DataFrame(y_original=temp_test[!,:Total_Trans_Amt],y_pred=predict(linreg,temp_test))
        perf_test.hata=perf_test[!,:y_original]-perf_test[!,:y_pred]
        println("$i. set için ortalama hata: ",mean(abs.(perf_test.hata)))
        hata=push!(hata,mean(abs.(perf_test.hata)))
    end
    println("Ortalama Hata: ",mean(hata))
end
```

Out[107]:
```
cross_validation (generic function with 1 method)
```

In [108]:
```
size(df)
```

Out[108]:
```
(10127, 10)
```

In [109]:
```
?collect;
```

In [110]:
```
a=collect(Kfold(size(df)[1],10))
a[5]
#temp_test=ny[setdiff(1:end,a[1]),:]

#temp_train=ny[a[1],:]
```

Out[110]:
```
9114-element Vector{Int64}:
     1
     2
     4
     5
     6
     7
     9
    10
    11
    12
    13
    14
    15
     ⋮
 10114
 10115
 10116
 10117
```

In [111]:
```
fm
```

Out[111]:
```
FormulaTerm
Response:
  Total_Trans_Amt(unknown)
Predictors:
  Total_Trans_Ct(unknown)
```

In [112]:
```
cross_validation(df,10,fm)
```

```
1. set için ortalama hata: 1565.1914608983016
2. set için ortalama hata: 1503.7644521417167
3. set için ortalama hata: 1451.6382328849074
4. set için ortalama hata: 1489.7861719572945
5. set için ortalama hata: 1466.496768505891
6. set için ortalama hata: 1455.1506093989055
7. set için ortalama hata: 1439.7954321217794
8. set için ortalama hata: 1465.0219163625204
9. set için ortalama hata: 1424.4403698299895
10. set için ortalama hata: 1486.9841492098928
Ortalama Hata: 1474.82695633112
```

SONUÇ

- Ortalama Mutlak Test Hatası: 1480.7757594744112
- Ortalama Mutlak Yüzde Test Hatası: 0.4068214071100352
- Kök Ortalama Kare Test Hatası: 2003.3479204355567
- Hata Kareleri Ortalaması Test Hatası: 4.01340289031347e6

Ortalama Mutlak Test Hatası (MAE) Düşük MAE daha iyi bir model performansını gösterir.

MAE, modelin tahminlerinin gerçek değerlerden ortalama olarak ne kadar uzak olduğunu ölçer. Bu durumda, ortalama olarak, modelin tahminleri gerçek değerlerden 1480.78 birim uzaklıktadır.

Ortalama Mutlak Yüzde Test Hatası (MAPE) Düşük MAPE değerleri, daha iyi bir model performansını gösterir.

MAPE, tahmin hatalarının yüzde cinsinden ortalama büyüklüğünü ölçer. Bu durumda, modelin tahminleri genellikle %0.41 oranında bir hata yapmaktadır.

Kök Ortalama Kare Test Hatası (RMSE) Düşük RMSE değerleri, daha iyi bir model performansını gösterir.

RMSE, modelin tahminlerinin gerçek değerlerden ortalama olarak ne kadar uzak ve bu uzaklıkların karelerinin ortalama karekökünü ölçer. Bu durumda, ortalama olarak, modelin tahminleri gerçek değerlerden 2003.35 birim uzaklıktadır.

Hata Kareleri Ortalaması Test Hatası (MSE) Bu değer ne kadar düşükse, modelin tahminleri o kadar iyi kabul edilir.

MSE, tahmin hatalarının karelerinin ortalamasını ölçer. Bu durumda, hataların karelerinin ortalaması 4.01e6(yani çok küçük değer).

Sonuçlar genel olarak modelin kabul edilebilir bir performansa sahip olduğunu göstermektedir.

Çoklu Doğrusal Regresyon

Bir bağımlı değişkenin birden fazla bağımsız değişken tarafından açıklanmaya çalışıldığı bir regresyon modelidir.

DATA

Bank Churners(Kredi Kartı Müşterileri) kullanmaya devam edebiliriz. Datatype olarak da uygundur hem kategorik hem numerik değerleri içerir. Total_Trans_Amt (toplam transfer tutarı) bağımlı değişken, Total_Trans_Ct (müşterinin işlem sayısı) , Credit_Limit , Months_on_book değişkenlerimizi bağımsız değişken olarak düşünerek işlemlerimizi sürdüreceğiz.

In [113]:
```
using GLM
fm1=@formula(Total_Trans_Amt ~ Total_Trans_Ct + Credit_Limit + Months_on_book)
linreg1=lm(fm1,train)
```

Out[113]:
```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgebra.CholeskyPivoted{Float64, Matrix{Float64}}}}, Matrix{Float64}}

Total_Trans_Amt ~ 1 + Total_Trans_Ct + Credit_Limit + Months_on_book

Coefficients:
────────────────────────────────────────────────────────────────────────────────────
                    Coef.   Std. Error       t  Pr(>|t|)    Lower 95%    Upper 95%
────────────────────────────────────────────────────────────────────────────────────
(Intercept)     -3480.66    123.936      -28.08    <1e-99  -3723.61     -3237.71
Total_Trans_Ct    116.196     0.958341   121.25    <1e-99    114.318      118.075
Credit_Limit        0.043852  0.00248101  17.68    <1e-67      0.0389885    0.0487154
Months_on_book     -0.594068  2.83252     -0.21    0.8339     -6.14659      4.95845
────────────────────────────────────────────────────────────────────────────────────
```

In [114]:
```
r2(linreg1)
```

Out[114]:
```
0.6694107579115809
```

In [115]:
```
test_pred1=predict(linreg1,test)
train_pred1=predict(linreg1,train)
```

Out[115]:
```
7651-element Vector{Union{Missing, Float64}}:
  1932.937338284392
   689.7169434665677
 -1028.2380698388938
 -1031.6543890621076
   -32.836875435454644
  1607.6886272432594
   266.84005836338315
   727.3698526607343
  1677.0777003377093
   -92.80598565692688
 -1011.4085825565217
   710.794817258449
  -179.6749008241357
     ⋮
  6957.9974714198715
  7871.373718173407
  9410.627063324906
 10090.210512471254
```

In [116]:
```
perf_test1=DataFrame(y_original=test[!,:Total_Trans_Amt],y_pred=test_pred1)
```

Out[116]: 2,476 rows × 2 columns

	y_original	y_pred
	Int64	Float64?
1	1088	-537.493
2	1538	1961.62
3	1570	-32.6629
4	1207	-427.984
5	692	-1587.56
6	1197	-187.169
7	1045	1119.68
8	1407	1648.75
9	1464	-95.2522
10	704	-1309.15

In [117]:
```
perf_test1.hata=perf_test1[!,:y_original]-perf_test1[!,:y_pred]
```

Out[117]:
```
2476-element Vector{Float64}:
 1625.4925453754013
 -423.62199045083776
 1602.662887220191
 1634.9840127298075
 2279.559841686032
 1384.1685286593543
  -74.67639249028912
 -241.74953229192624
 1559.2522273659254
 2013.149460483859
 1369.0768858124843
 1938.6243540552525
 1669.6040003040725
    ⋮
 5052.071255463949
 2903.463309502775
 3726.062922272662
 5957.933261404007
```

In [118]:
```
perf_test1.hata_kare=perf_test1.hata.*perf_test1.hata
```

Out[118]:
```
2476-element Vector{Float64}:
      2.642226015071001e6
 179455.59079352967
      2.5685283300729585e6
      2.6731727218820634e6
      5.196393071827647e6
      1.9159225157310015e6
   5576.563595363709
  58442.83636336509
      2.4312675085455994e6
      4.0527707502464526e6
      1.8743715192660103e6
      3.758264386136145e6
      2.7875775178313614e6
      ⋮
      2.5523423970285077e7
      8.430099189628806e6
      1.388354490073509e7
      3.549696874734419e7
```

In [119]:
```
perf_test1
```

Out[119]: 2,476 rows × 4 columns

	y_original	y_pred	hata	hata_kare
	Int64	Float64?	Float64	Float64
1	1088	-537.493	1625.49	2.64223e6
2	1538	1961.62	-423.622	1.79456e5
3	1570	-32.6629	1602.66	2.56853e6
4	1207	-427.984	1634.98	2.67317e6
5	692	-1587.56	2279.56	5.19639e6
6	1197	-187.169	1384.17	1.91592e6
7	1045	1119.68	-74.6764	5576.56
8	1407	1648.75	-241.75	58442.8
9	1464	-95.2522	1559.25	2.43127e6
10	704	-1309.15	2013.15	4.05277e6

In [120]:
```
perf_train1=DataFrame(y_original=train[!,:Total_Trans_Amt],y_pred=train_pred)
```

Out[120]: 7,651 rows × 2 columns

	y_original	y_pred
	Int64	Float64?
1	1144	1727.2
2	1291	669.031
3	1887	-859.43
4	1171	-859.43
5	816	81.1616
6	1330	433.883
7	1350	-389.134
8	1441	551.457
9	1201	1727.2
10	1314	-153.986

In [121]:
```
perf_train1.hata=perf_train[!,:y_original]-perf_train[!,:y_pred]
```

Out[121]:
```
7651-element Vector{Float64}:
 -583.1966234217434
  621.968754345995
 2746.429855566062
 2030.429855566062
  734.8384086614055
  896.1166160721596
 1739.1341321137338
  889.5426852090773
 -526.1966234217434
 1467.9862703875697
 2751.1516481553085
  641.968754345995
 1384.4123395244878
    ⋮
 3436.1243494657274
 6646.8111099718235
 7023.350008751757
 5396.628216162509
```

In [122]:
```
perf_train1.hata_kare=perf_train1.hata.*perf_train1.hata
```

Out[122]:
```
7651-element Vector{Float64}:
 340118.30157052283
 386845.1313827086
      7.542876951544621e6
      4.1226453983740197e6
 539987.4868440268
 803024.9896006183
      3.02458752948299e6
 791286.1888089755
 276882.8865004441
      2.154983690046407e6
      7.56883539114767e6
 412123.8815565484
      1.9165975258276658e6
      ⋮
      1.1806950544991268e7
      4.4180097931644864e7
      4.93274453454333e7
      2.912359610348135e7
```

In [123]:
```
perf_train1
```

Out[123]: 7,651 rows × 4 columns

	y_original	y_pred	hata	hata_kare
	Int64	Float64?	Float64	Float64
1	1144	1727.2	-583.197	3.40118e5
2	1291	669.031	621.969	3.86845e5
3	1887	-859.43	2746.43	7.54288e6
4	1171	-859.43	2030.43	4.12265e6
5	816	81.1616	734.838	5.39987e5
6	1330	433.883	896.117	803025.0
7	1350	-389.134	1739.13	3.02459e6
8	1441	551.457	889.543	7.91286e5
9	1201	1727.2	-526.197	2.76883e5
10	1314	-153.986	1467.99	2.15498e6

Test Hatası Karşılaştırması

In [124]:
```
println("Ortalama Mutlak Test Hatası: ",mean(abs.(perf_test1.hata)))
println("Ortalama Mutlak Yüzde Test Hatası: ",mape(perf_test1))
println("Kök Ortalama Kare Test Hatası: ", rmse(perf_test1))
println("Hata Kareleri Ortalaması Test Hatası: ", mean(perf_test1.hata_kare))
```

```
Ortalama Mutlak Test Hatası: 1461.0795718869274
Ortalama Mutlak Yüzde Test Hatası: 0.40901746894871704
Kök Ortalama Kare Test Hatası: 1980.8976208599981
Hata Kareleri Ortalaması Test Hatası: 3.9239553843288007e6
```

Test Hata Dağılımı

In [125]:
```
histogram(perf_test1.hata,title="Test Hata Analizi",ylabel="Frekans",xlabel="Hata",legend=true)
```

Out[125]:

Train Hataları Karşılaştırması

In [126]:
```
println("Ortalama Mutlak Train Hatası: ",mean(abs.(perf_train1.hata)))
println("Ortalama Mutlak Yüzde Train Hatası: ",mape(perf_train1))
println("Kök Ortalama Kare Train Hatası: ", rmse(perf_train1))
println("Hata Kareleri Ortalaması Train Hatası: ", mean(perf_train1.hata_kare))
```

```
Ortalama Mutlak Train Hatası: 1481.400473977008
Ortalama Mutlak Yüzde Train Hatası: 0.40606830591318527
Kök Ortalama Kare Train Hatası: 2005.9244050795019
Hata Kareleri Ortalaması Train Hatası: 4.0237327188935536e6
```

Train Hata Dağılımı

In [127]:
```
histogram(perf_train1.hata,title="Train Hata Analizi",ylabel="Frekans",xlabel="Hata",legend=true)
```

Out[127]:

In [128]:
```
cross_validation(df,10,fm1)
```

```
1. set için ortalama hata: 1470.3865226829723
2. set için ortalama hata: 1436.098006232669
3. set için ortalama hata: 1379.7908007324522
4. set için ortalama hata: 1476.3972056287175
5. set için ortalama hata: 1481.9179146600916
6. set için ortalama hata: 1414.5474453935733
7. set için ortalama hata: 1481.0364524739136
8. set için ortalama hata: 1440.2364378282584
9. set için ortalama hata: 1462.4085557673973
10. set için ortalama hata: 1470.8328318425158
Ortalama Hata: 1451.365217324256
```

Sonuç

- Ortalama Mutlak Test Hatası: 1461.0795718869274
- Ortalama Mutlak Yüzde Test Hatası: 0.40901746894871704
- Kök Ortalama Kare Test Hatası: 1980.8976208599981
- Hata Kareleri Ortalaması Test Hatası: 3.9239553843288007e6
- Ortalama Mutlak Train Hatası: 1481.400473977008
- Ortalama Mutlak Yüzde Train Hatası: 0.40606830591318527
- Kök Ortalama Kare Train Hatası: 2005.9244050795019
- Hata Kareleri Ortalaması Train Hatası: 4.0237327188935536e6

- Ortalama Mutlak Test Hatası (MAE): 1461.08
- Ortalama Mutlak Yüzde Test Hatası (MAPE): 0.409
- Kök Ortalama Kare Test Hatası (RMSE): 1980.90
- Hata Kareleri Ortalaması Test Hatası (MSE): 3.92e6
- Ortalama Mutlak Train Hatası (MAE): 1481.40
- Ortalama Mutlak Yüzde Train Hatası (MAPE): 0.406
- Kök Ortalama Kare Train Hatası (RMSE): 2005.92
- Hata Kareleri Ortalaması Train Hatası (MSE): 4.02e6

Düşük MAE, daha iyi bir tahmin performansını ifade eder. Düşük MAPE, daha iyi bir model performansını ifade eder. Burada %0.409 oranında bir ortalama hata oranı var. Düşük RMSE, daha iyi bir tahmin performansını ifade eder. Düşük MSE, daha iyi bir model performansını ifade eder.

Sonuçlar genel olarak modelin kabul edilebilir iyi bir performansa sahip olduğunu göstermektedir.

SINIFLANDIRMA MODELLERİ

Sınıflandırma modelleri, bir veri örneğini belirli bir kategoriye (sınıfa) atayan modellerdir. Bu tür modeller, genellikle örneğin belirli bir etiket veya sınıf içindeki kategorisini tahmin etmek amacıyla kullanılır. Sınıflandırma modelleri, birçok uygulama alanında yaygın olarak kullanılır ve önemli bir makine öğrenimi türüdür.

Lojistik Regresyon

Temelde bir doğrusal regresyon modelini sınıflandırma problemlerine uyarlar. Binary (iki sınıflı) ve çok sınıflı sınıflandırma problemlerinde kullanılabilir.

Veri Adı: Kalp Datası Bu veriyi seçmemizin nedeni output'u ikili sınıflandırma olduğu için (kalp krizi riski az ve fazla olarak) lojistik regresyon modeline uyumluluk sağlar.

Veri Kaynağı: https://github.com/Ahmet-SANCAKLI/Python-ile-Veri-Analizi-ve-Makine-Ogrenmesi-Projesi/blob/main/heart.csv

Veri Setinin Açıklaması

Age : Hastaların yaşı
Sex : Hastaların cinsiyeti
cp : Göğüs ağrısı türü
Value 1: typical angina
Value 2: atypical angina
Value 3: non-anginal pain
Value 4: asymptomatic
trtbps : dinlenme kan basıncı (in mm Hg)
chol : kolesterol mg/dl fetched via BMI sensor
fbs : (açlık kan şekeri > 120 mg/dl) (1 = true; 0 = false)
restecg : dinlenme elektrokardiyografik sonuçları
Value 0: normal
Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
thalachh : ulaşılan maksimum kalp hızı
exang: egzersiz kaynaklı anjina(1 = yes; 0 = no)
oldpeak : öncelikli zirve
slp : eğim
caa : Number of major vessels
thall : Talyum Stres Testi sonucu (0,3)
output : 0 = kalp krizi riski daha az, 1 = kalp krizi riski daha fazla

In [2]:
```julia
using Pkg
Pkg.add("DataFrames")
Pkg.add("CSV")
Pkg.add("Statistics")
Pkg.add("MLDataUtils")
Pkg.add("Impute")
Pkg.add("Plots")
```

```
   Updating registry at `C:\Users\Kev\.julia\registries\General`
   Updating git-repo `https://github.com/JuliaRegistries/General.git`
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
 No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
```

In [3]:
```julia
using CSV
using DataFrames
df2=CSV.read("C:/Users/Kev/Desktop/heart.csv",DataFrame)
```

Out[3]: 303 rows × 14 columns (omitted printing of 3 columns)

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp
	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	String3	Int64
1	63	1	3	145	233	1	0	150	0	2,3	0
2	37	1	2	130	250	0	1	187	0	3,5	0
3	41	0	1	130	204	0	0	172	0	1,4	2
4	56	1	1	120	236	0	1	178	0	0,8	2
5	57	0	0	120	354	0	1	163	1	0,6	2
6	57	1	0	140	192	0	1	148	0	0,4	1
7	56	0	1	140	294	0	0	153	0	1,3	1
8	44	1	1	120	263	0	1	173	0	0,0	2
9	52	1	2	172	199	1	1	162	0	0,5	2
10	57	1	2	150	168	0	1	174	0	1,6	2

Datamızda oldpeak string olarak gözükmekte onu kendi type 'ı olan float'a çevirelim.

In [4]:
```
df2.oldpeak = parse.(Float64, replace.(df2.oldpeak, "," => "."))
```

Out[4]:
```
303-element Vector{Float64}:
 2.3
 3.5
 1.4
 0.8
 0.6
 0.4
 1.3
 0.0
 0.5
 1.6
 1.2
 0.2
 0.6
 ⋮
 4.4
 2.8
 0.8
 2.8
```

In [5]:
```
describe(df2) #tekrar inceleyelim.
```

Out[5]: 14 rows × 8 columns

	variable	mean	min	median	max	nunique	nmissing	eltype
	Symbol	Float64	Real	Float64	Real	Nothing	Nothing	DataType
1	age	54.3663	29	55.0	77			Int64
2	sex	0.683168	0	1.0	1			Int64
3	cp	0.966997	0	1.0	3			Int64
4	trtbps	131.624	94	130.0	200			Int64
5	chol	246.264	126	240.0	564			Int64
6	fbs	0.148515	0	0.0	1			Int64
7	restecg	0.528053	0	1.0	2			Int64
8	thalachh	149.647	71	153.0	202			Int64
9	exng	0.326733	0	0.0	1			Int64
10	oldpeak	1.0396	0.0	0.8	6.2			Float64

In [6]:
```
size(df2)
```

Out[6]: (303, 14)

In [7]:
```
using Lathe.preprocess:TrainTestSplit
train,test=TrainTestSplit(df2,.70)
#Verinin %70'inin eğitim veri setine ayrılacağını belirtelim. Yani, geriye kalan %30'u test veri setine ayrıldı.
```

Out[7]:
```
(211×14 DataFrame. Omitted printing of 6 columns
│ Row │ age   │ sex   │ cp    │ trtbps │ chol  │ fbs   │ restecg │ thalachh │
│     │ Int64 │ Int64 │ Int64 │ Int64  │ Int64 │ Int64 │ Int64   │ Int64    │
├─────┼───────┼───────┼───────┼────────┼───────┼───────┼─────────┼──────────┤
│ 1   │ 37    │ 1     │ 2     │ 130    │ 250   │ 0     │ 1       │ 187      │
│ 2   │ 41    │ 0     │ 1     │ 130    │ 204   │ 0     │ 0       │ 172      │
│ 3   │ 56    │ 1     │ 1     │ 120    │ 236   │ 0     │ 1       │ 178      │
│ 4   │ 57    │ 0     │ 0     │ 120    │ 354   │ 0     │ 1       │ 163      │
│ 5   │ 57    │ 1     │ 0     │ 140    │ 192   │ 0     │ 1       │ 148      │
│ 6   │ 44    │ 1     │ 1     │ 120    │ 263   │ 0     │ 1       │ 173      │
│ 7   │ 52    │ 1     │ 2     │ 172    │ 199   │ 1     │ 1       │ 162      │
│ 8   │ 57    │ 1     │ 2     │ 150    │ 168   │ 0     │ 1       │ 174      │
│ 9   │ 54    │ 1     │ 0     │ 140    │ 239   │ 0     │ 1       │ 160      │
│ 10  │ 48    │ 0     │ 2     │ 130    │ 275   │ 0     │ 1       │ 139      │
⋮
│ 201 │ 57    │ 1     │ 0     │ 110    │ 335   │ 0     │ 1       │ 143      │
│ 202 │ 61    │ 1     │ 0     │ 148    │ 203   │ 0     │ 1       │ 161      │
│ 203 │ 58    │ 1     │ 0     │ 114    │ 318   │ 0     │ 2       │ 140      │
│ 204 │ 58    │ 0     │ 0     │ 170    │ 225   │ 1     │ 0       │ 146      │
```

In [8]:
```
using Pkg
Pkg.status()
```

```
      Status `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  [336ed68f] CSV v0.10.11
  [324d7699] CategoricalArrays v0.8.3
  [8f4d0f93] Conda v1.10.0
  [a93c6f00] DataFrames v0.21.8
  [31c24e10] Distributions v0.25.24
  [38e38edf] GLM v1.9.0
  [09f84164] HypothesisTests v0.10.13
  [7073ff75] IJulia v1.24.2
  [f7bf1975] Impute v0.6.11
  [5ab0869b] KernelDensity v0.6.8
  [38d8eb38] Lathe v0.1.3
  [f0e99cf1] MLBase v0.8.0
  [cc2ba9b6] MLDataUtils v0.5.4
  [91a5bcdd] Plots v1.39.0
  [438e738f] PyCall v1.96.4
  [ce6b1742] RDatasets v0.7.7
  [f535d66d] ROCAnalysis v0.3.6
  [2913bbd2] StatsBase v0.33.21
```

In [9]:
```
using DataFrames
using GLM
```

Build Model - Model Kurma

In [10]:
```
fm =@formula(output~age+sex+cp+trtbps+chol+fbs+restecg+thalachh+exng+oldpeak+slp+caa+thall)
logit=glm(fm,train,Binomial(),LogitLink())
```

Out[10]:
```
StatsModels.TableRegressionModel{GeneralizedLinearModel{GLM.GlmResp{Vector{Float64}, Binomial{Float64}, LogitLink}, GLM.Dens
ePredChol{Float64, LinearAlgebra.CholeskyPivoted{Float64, Matrix{Float64}}}}, Matrix{Float64}}

output ~ 1 + age + sex + cp + trtbps + chol + fbs + restecg + thalachh + exng + oldpeak + slp + caa + thall

Coefficients:
──────────────────────────────────────────────────────────────────────────────
                    Coef.  Std. Error      z  Pr(>|z|)   Lower 95%   Upper 95%
──────────────────────────────────────────────────────────────────────────────
(Intercept)   3.73778     3.16027      1.18    0.2369  -2.45624     9.9318
age          -0.0198679   0.0296993   -0.67    0.5035  -0.0780774   0.0383416
sex          -1.75906     0.536845    -3.28    0.0011  -2.81126    -0.706865
cp            0.818615    0.220365     3.71    0.0002   0.386708    1.25052
trtbps       -0.013688    0.0125572   -1.09    0.2757  -0.0382998   0.0109237
chol         -0.00578949  0.00438316  -1.32    0.1866  -0.0143803   0.00280134
fbs          -0.24252     0.593693    -0.41    0.6829  -1.40614     0.921098
restecg       0.345287    0.421974     0.82    0.4132  -0.481767    1.17234
thalachh      0.0236973   0.01257      1.89    0.0594  -0.0009395   0.0483341
exng         -1.51054     0.532115    -2.84    0.0045  -2.55347    -0.467616
```

In [11]:
```
using Plots
scatter(df2[!,:output])
```

Out[11]:

```
In [12]: using GLM
         predictions=predict(logit,test)
```

```
Out[12]: 92-element Vector{Union{Missing, Float64}}:
          0.8211929392973583
          0.8482127345053222
          0.8584239799233027
          0.6062941347010239
          0.8311585804132117
          0.8044368711635063
          0.8228614427755033
          0.9041555957726622
          0.7994936938645351
          0.4560694435986059
          0.6849392714709094
          0.9018349935696284
          0.9919567520562725
          ⋮
          0.27448287611091643
          0.09737834601907788
          0.486611834868704
          0.007286349190117717
```

```
In [13]: scatter(predictions, title="Tahmin Olasılık Değerleri",legend=false)
Out[13]:
```

```
In [14]: prediction_class=[if x<0.5 0 else 1 end for x in predictions]
```

```
Out[14]: 92-element Vector{Int64}:
          1
          1
          1
          1
          1
          1
          1
          1
          1
          0
          1
          1
          1
          ⋮
          0
          0
          0
          0
```

```
In [15]: scatter(test[!,:output], title="Gerçek")
Out[15]:
```

```
In [16]: scatter(prediction_class, opacity=0.3, title="Tahmin")
Out[16]:
```


```
In [16]: scatter(prediction_class, opacity=0.3, title="Tahmin")
Out[16]:
```


```
In [17]: prediction_df2=DataFrame(y_actual=test.output ,prob_predicted=predictions,y_predicted=prediction_class)
```

Out[17]: 92 rows × 3 columns

	y_actual	prob_predicted	y_predicted
	Int64	Float64?	Int64
1	1	0.821193	1
2	1	0.848213	1
3	1	0.858424	1
4	1	0.606294	1
5	1	0.831159	1
6	1	0.804437	1
7	1	0.822861	1
8	1	0.904156	1
9	1	0.799494	1
10	1	0.456069	0

```
In [18]: using MLBase
```

```
In [19]: confusion_matrix=MLBase.roc(prediction_df2.y_actual,prediction_df2.y_predicted)
```

```
Out[19]: ROCNums{Int64}
           p = 49
           n = 43
           tp = 41
           tn = 37
           fp = 6
           fn = 8
```

In [18]:
```
using MLBase
```

In [19]:
```
confusion_matrix=MLBase.roc(prediction_df2.y_actual,prediction_df2.y_predicted)
```

Out[19]:
```
ROCNums{Int64}
  p = 49
  n = 43
  tp = 41
  tn = 37
  fp = 6
  fn = 8
```

In [20]:
```
accuracy(cm)=(cm.tp+cm.tn)/(cm.tp+cm.tn+cm.fp+cm.fn)
```

Out[20]:
```
accuracy (generic function with 1 method)
```

In [21]:
```
specificity(cm)=cm.tn/(cm.tn+cm.fp)
```

Out[21]:
```
specificity (generic function with 1 method)
```

In [22]:
```
recall(cm)=cm.tp/(cm.tp+cm.fn)
```

Out[22]:
```
recall (generic function with 1 method)
```

In [23]:
```
F1score(cm)=2*cm.tp/(2*cm.tp+cm.fp+cm.fn)
```

Out[23]:
```
F1score (generic function with 1 method)
```

In [24]:
```
println("Accuracy: ",accuracy(confusion_matrix))
println("Specificity: ",specificity(confusion_matrix))
println("Precision: ",precision(confusion_matrix))
println("Recall: ",recall(confusion_matrix))
println("F1Score: ",F1score(confusion_matrix))
```

```
Accuracy: 0.8478260869565217
Specificity: 0.8604651162790697
Precision: 0.8723404255319149
Recall: 0.8367346938775511
F1Score: 0.8541666666666666
```

Sonuç

- Accuracy: 0.8478260869565217
- Specificity: 0.8604651162790697
- Precision: 0.8723404255319149
- Recall: 0.8367346938775511
- F1Score: 0.8541666666666666

Sonuçlara göre,

Accuracy(Doğruluk): Modelin doğruluk oranı oldukça yüksektir(0.85).

Specificity (Özgüllük): Model, negatif sınıfları (genellikle "negatif" olarak etiketlenenleri) tanımlamada oldukça başarılıdır (0.86). Bu, modelin gerçek negatifleri kaçırma olasılığının düşük olduğu anlamına gelir.

Recall (Duyarlılık): Model, gerçek pozitifleri (gerçekten "pozitif" olanları) yakalama konusunda ortalama bir performans sergiliyor (0.84). Bu, modelin pozitif örnekleri kaçırma eğiliminde olduğu anlamına gelir.

Precision (Hassasiyet): Modelin pozitif olarak tahmin ettiği örneklerin gerçekten pozitif olma olasılığı oldukça yüksektir(0.87).

F1 Score: Hassasiyet ve duyarlılık dengesi olarak düşünülen F1 skoru, 0.85 olarak oldukça sağlıklı bir denge gösteriyor.

Sonuç olarak, modelin genel olarak iyi bir performans sergilediği söylenebilir. Genelde akademi metinlerinde modeli yorumlarken Accuracy ve F1 Scorelarına bakılır, ve ikisininde birbirine yakın değerler çıkması beklenir. Modelimiz %85 düzeyinde başarılıdır denilir.

Karar Ağaçları(CART)

Ağaç yapısını kullanarak veri kümesini belirli sınıflara böler. Karar ağaçları, açıklanabilirlikleri ve yüksek performansları nedeniyle popülerdir. Random Forest ise birçok karar ağacını bir araya getirerek daha güçlü bir sınıflandırma sağlar.

Veri adı: Air pollution - Hava kirliliği

Veri Tanıtımı

Veri seti, 133 ülkedeki 6.985 şehrin PM2.5 hava kalitesi verilerini içeriyor. Veriler, 30.000'den fazla düzenleyici hava kalitesi izleme istasyonundan ve düşük maliyetli hava kalitesi sensörlerinden toplandı. Bu izleme istasyonları ve sensörler, dünya çapındaki devlet kurumları, araştırma kurumları, kar amacı gütmeyen sivil toplum kuruluşları, üniversiteler ve eğitim tesisleri, özel şirketler ve vatandaş bilim adamları tarafından işletilmektedir. PM2.5 verileri metreküp başına mikrogram (µg/m³) cinsinden ölçülür.

Krar ağaçları, kategorik ve sayısal verilerle çalışabilen esnek bir model olduğundan hava kirliliği üzerinde çalışmak istedim.

Data Kaynağı: https://www.kaggle.com/datasets/andreinovikov/air-pollution/data

city: Name of the city
country: Name of the country
2017: Average PM2.5 concentration in 2017
2018: Average PM2.5 concentration in 2018
2019: Average PM2.5 concentration in 2019
2020: Average PM2.5 concentration in 2020
2021: Average PM2.5 concentration in 2021
2022: Average PM2.5 concentration in 2022
2023: Average PM2.5 concentration in 2023

Verimizde eksik datalar olduğundan sadeleştirmeye gidilmiştir.2021-2022-2023 verileri kullanma kararı alınmıştır.

In [48]:
```
using CSV
using DataFrames

# Doğru dosya yolunu belirtin
csv_dosya_yolu = "C:/Users/Kev/Desktop/air_pollution.csv"

# CSV dosyasını DataFrame'e yükle
air = DataFrame(CSV.File(csv_dosya_yolu))
```

Out[48]: 6,199 rows × 5 columns

Out[48]: 6,199 rows × 5 columns

	city	country	2021Y	2022Y	2023Y
	String	String	Int64	Int64	Int64
1	Kabul	Afghanistan	38	17	18
2	Tirana	Albania	13	15	14
3	Algiers	Algeria	20	18	17
4	Ordino	Andorra	7	5	5
5	Luanda	Angola	11	9	9
6	Buenos Aires	Argentina	14	14	14
7	Cordoba	Argentina	10	9	9
8	General Pico	Argentina	7	7	7
9	Mendoza	Argentina	9	8	8
10	Rafaela	Argentina	10	11	11

In [49]:
```
air = select(air, Not(:city))
```

Out[49]: 6,199 rows × 4 columns

	country	2021Y	2022Y	2023Y
	String	Int64	Int64	Int64
1	Afghanistan	38	17	18
2	Albania	13	15	14
3	Algeria	20	18	17
4	Andorra	7	5	5
5	Angola	11	9	9
6	Argentina	14	14	14
7	Argentina	10	9	9
8	Argentina	7	7	7
9	Argentina	9	8	8
10	Argentina	10	11	11

In [50]:
```
describe(air)
```

Out[50]: 4 rows × 8 columns (omitted printing of 1 columns)

	variable	mean	min	median	max	nunique	nmissing
	Symbol	Union...	Any	Union...	Any	Union...	Nothing
1	country		Afghanistan		Zambia	129	
2	2021Y	1.07379e14	2	11.0	7899999999999900		
3	2022Y	2.37175e14	1	9.0	7899999999999900		
4	2023Y	2.12129e14	1	9.0	7899999999999900		

In [51]:
```
using Lathe.preprocess: TrainTestSplit
train,test=TrainTestSplit(air,.60)
```

Out[51]:
```
(3662×4 typename(DataFrame)
│ Row  │ country     │ 2021Y │ 2022Y │ 2023Y │
│      │ String      │ Int64 │ Int64 │ Int64 │
├──────┼─────────────┼───────┼───────┼───────┤
│ 1    │ Afghanistan │ 38    │ 17    │ 18    │
│ 2    │ Albania     │ 13    │ 15    │ 14    │
│ 3    │ Algeria     │ 20    │ 18    │ 17    │
│ 4    │ Andorra     │ 7     │ 5     │ 5     │
│ 5    │ Angola      │ 11    │ 9     │ 9     │
│ 6    │ Argentina   │ 10    │ 9     │ 9     │
│ 7    │ Argentina   │ 7     │ 7     │ 7     │
│ 8    │ Argentina   │ 9     │ 8     │ 8     │
│ 9    │ Armenia     │ 37    │ 28    │ 28    │
│ 10   │ Australia   │ 5     │ 4     │ 4     │
⋮
│ 3652 │ Uzbekistan  │ 43    │ 34    │ 32    │
│ 3653 │ Venezuela   │ 5     │ 4     │ 5     │
│ 3654 │ Vietnam     │ 6     │ 4     │ 4     │
│ 3655 │ Vietnam     │ 14    │ 19    │ 19    │
```

In [52]:
```
# Feature selection (Train)
features = Array(train[:, [Symbol("2021Y"), Symbol("2022Y"), Symbol("2023Y")]])
labels = Array(train[:,:country])

# Convert features to float
features = float.(features)
labels = string.(labels)
```

Out[52]:
```
3662-element Vector{String}:
 "Afghanistan"
 "Albania"
 "Algeria"
 "Andorra"
 "Angola"
 "Argentina"
 "Argentina"
 "Argentina"
 "Armenia"
 "Australia"
 "Australia"
 "Australia"
 "Australia"
 ⋮
 "Uruguay"
 "Uzbekistan"
 "Venezuela"
 "Vietnam"
```

In [53]:
```
# feature_selection (Test)

features_test = Array(train[:, [Symbol("2021Y"), Symbol("2022Y"), Symbol("2023Y")]])
labels_test = Array(train[:,:country])
features_test = float.(features_test)
labels_test = string.(labels_test)
```

Out[53]:
```
3662-element Vector{String}:
 "Afghanistan"
 "Albania"
 "Algeria"
 "Andorra"
 "Angola"
 "Argentina"
 "Argentina"
 "Argentina"
 "Armenia"
 "Australia"
 "Australia"
 "Australia"
 "Australia"
 ⋮
 "Uruguay"
 "Uzbekistan"
 "Venezuela"
 "Vietnam"
```

Birinci Yol

In [54]:
```
using Pkg
Pkg.add("DecisionTree")
Pkg.add("MLBase")
using DecisionTree
using MLBase
Pkg.update()
```

```
  Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 2 seconds (162 already precompiled, 6 skipped during auto due to previous error
s)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
  Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 2 seconds (162 already precompiled, 6 skipped during auto due to previous error
s)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Updating registry at `C:\Users\Kev\.julia\registries\General`
   Updating git-repo `https://github.com/JuliaRegistries/General.git`
  Installed StatsPlots ──────────── v0.10.2
  Installed CategoricalArrays ───── v0.5.5
  Installed Plots ───────────────── v0.16.0
  Installed FileIO ──────────────── v1.5.1
  Installed NearestNeighbors ────── v0.4.3
```

In [55]:
```
Pkg.add("MLJ")
using MLJ
```

```
  Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
Precompiling project...
  ✓ Requires
  ✓ Ratios
  ✓ OffsetArrays
  ✓ RecipesBase
  ✓ MappedArrays
  ✓ Wayland_protocols_jll
  ✓ TableTraitsUtils
  ✓ FixedPointNumbers
  ✗ PDMats
  ✓ Libtiff_jll
  ✓ EzXML
  ✓ CategoricalArrays
  ✓ StaticArrays
  ✓ MLLabelUtils
  ✓ xkbcommon_jll
```

In [56]:
```
using DecisionTree #fit fonksiyonu çalıştıramadığım için google yardımıyla farklı yapıda kullanmak zorunda kaldım.

model = build_tree(labels, features)
```

Out[56]:
```
Decision Tree
Leaves: 742
Depth:  17
```

In [57]:
```
# Modeli göster
show(model)
```

```
Decision Tree
Leaves: 742
Depth:  17
```

In [58]:
```
print_tree(model, 17) #derinliğim 17 çıktığı için kökten başlayarak 21 seviyeye kadar olan karar ağaç yapısını yazdıracak.
```

```
Feature 3 < 13.5 ?
├─ Feature 3 < 7.5 ?
    ├─ Feature 1 < 6.5 ?
        ├─ Feature 2 < 4.5 ?
            ├─ Feature 3 < 3.5 ?
                ├─ Feature 1 < 3.5 ?
                    ├─ Feature 2 < 1.5 ?
                        ├─ Feature 1 < 2.5 ?
                            ├─ Guam : 1/2
                            └─ USA : 1/1
                        └─ Feature 1 < 2.5 ?
                            ├─ Feature 3 < 2.5 ?
                                ├─ Australia : 6/8
                                └─ Australia : 2/2
                            └─ Feature 3 < 2.5 ?
                                ├─ Australia : 6/8
                                └─ Australia : 3/7
                    └─ Feature 1 < 4.5 ?
                        ├─ Feature 2 < 2.5 ?
```

```
In [59]: # Test verisi üzerinde çalıştıralım.
         predictions = apply_tree(model, features_test)

Out[59]: 3662-element Vector{String}:
          "Afghanistan"
          "Albania"
          "Algeria"
          "USA"
          "USA"
          "USA"
          "USA"
          "USA"
          "Armenia"
          "Canada"
          "USA"
          "USA"
          "USA"
          ⋮
          "USA"
          "Uzbekistan"
          "Venezuela"
          "USA"
```

```
In [60]: apply_tree_proba(model,[37.5,17.1,18.1] , [Symbol("2021Y"), Symbol("2022Y"), Symbol("2023Y")]) #ama bu seferde böyle hata verdi

KeyError: key "Afghanistan" not found

Stacktrace:
  [1] getindex(h::Dict{Symbol, Int64}, key::String)
    @ Base .\dict.jl:482
  [2] compute_probabilities(labels::Vector{Symbol}, votes::Vector{String}, weights::Float64)
    @ DecisionTree C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:16
  [3] compute_probabilities
    @ C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:12 [inlined]
  [4] apply_tree_proba
    @ C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:315 [inlined]
  [5] apply_tree_proba(tree::Node{Float64, String}, features::Vector{Float64}, labels::Vector{Symbol}) (repeats 3 times)
    @ DecisionTree C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:324
  [6] apply_tree_proba(tree::Node{Float64, String}, features::Vector{Float64}, labels::Vector{Symbol})
    @ DecisionTree C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:322
  [7] apply_tree_proba(tree::Node{Float64, String}, features::Vector{Float64}, labels::Vector{Symbol}) (repeats 3 times)
    @ DecisionTree C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:324
  [8] apply_tree_proba(tree::Node{Float64, String}, features::Vector{Float64}, labels::Vector{Symbol})
    @ DecisionTree C:\Users\Kev\.julia\packages\DecisionTree\0Dw1P\src\classification\main.jl:322
```

In [61]:
```
n_folds=3
accuracy = nfoldCV_tree(labels_test, features_test, n_folds) #Burada 3 kullanıldığında, veri kümeniz üç parçaya bölünecek ve mode
```

```
Fold 1
Classes:  ["Andorra", "Argentina", "Australia", "Austria", "Bangladesh", "Belgium", "Bosnia Herzegovina", "Brazil", "Bulgari
a", "Canada", "Chile", "China", "Colombia", "Croatia", "Czech Republic", "Democratic Republic of the Congo", "Denmark", "Egy
pt", "El Salvador", "Estonia", "Finland", "France", "Georgia", "Germany", "Ghana", "Greece", "Guam", "Guatemala", "Guyana",
"Honduras", "Hungary", "Iceland", "India", "Indonesia", "Iran", "Ireland", "Israel", "Italy", "Ivory Coast", "Japan", "Kazak
hstan", "Kosovo", "Kuwait", "Kyrgyzstan", "Lithuania", "Malaysia", "Malta", "Mexico", "Moldova", "Mongolia", "Montenegro",
"Nepal", "Netherlands", "New Zealand", "Nigeria", "North Macedonia", "Norway", "Pakistan", "Peru", "Philippines", "Poland",
"Portugal", "Qatar", "Romania", "Russia", "Saudi Arabia", "Serbia", "Slovakia", "Slovenia", "South Africa", "South Korea",
"Spain", "Sri Lanka", "Sudan", "Sweden", "Switzerland", "Syria", "Taiwan", "Tajikistan", "Thailand", "Trinidad and Tobago",
"Turkey", "USA", "Uganda", "Ukraine", "United Arab Emirates", "United Kingdom", "Venezuela", "Vietnam"]
Matrix:

89×89 Matrix{Int64}:
 0  0  0  0  0  0  0  0  0   0  0   0  …  0  0  0  0   1  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0   0  0   0     0  0  0  0   1  0  0  0  0  0  0
 0  0  5  0  0  0  1  0  0   3  0   0     0  0  0  0  15  0  0  0  7  0  0
 0  0  0  1  0  0  0  0  0   0  0   0     0  0  0  0   5  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0   0  0   0     0  0  0  0   0  0  0  0  0  0  0
```

In [62]:
```
# prune tree: merge leaves having >= 90% combined purity (default: 100%)
modelp = prune_tree(model, 0.6)
```

Out[62]:
```
Decision Tree
Leaves: 583
Depth:  17
```

In [63]:
```
# pretty print of the tree, to a depth of 5 nodes (optional)
print_tree(modelp, 17)
```

```
Feature 3 < 13.5 ?
├─ Feature 3 < 7.5 ?
    ├─ Feature 1 < 6.5 ?
        ├─ Feature 2 < 4.5 ?
            ├─ Feature 3 < 3.5 ?
                ├─ Feature 1 < 3.5 ?
                    ├─ Australia : 17/28
                    └─ Feature 1 < 4.5 ?
                        ├─ Feature 2 < 2.5 ?
                            ├─ USA : 4/7
                            └─ Feature 2 < 3.5 ?
                                ├─ New Zealand : 3/14
                                └─ Australia : 1/1
                        └─ Feature 1 < 5.5 ?
                            ├─ Feature 3 < 2.5 ?
                                ├─ Feature 2 < 2.5 ?
                                    ├─ Feature 3 < 1.5 ?
                                        ├─ Puerto Rico : 1/1
                                        └─ USA : 3/8
```

In [64]:
```
preds1=apply_tree(modelp,features_test)
# run 3-fold cross validation of pruned tree,
n_folds=3
accuracy1 = nfoldCV_tree(preds1, features_test, n_folds)
```

```
Fold 1
Classes:  ["Afghanistan", "Algeria", "Armenia", "Australia", "Austria", "Belgium", "Bosnia Herzegovina", "Brazil", "Bulgari
a", "Canada", "Chile", "China", "Colombia", "Croatia", "Czech Republic", "El Salvador", "Ethiopia", "France", "Georgia", "Ge
rmany", "Ghana", "Greece", "Hungary", "India", "Indonesia", "Iran", "Ireland", "Israel", "Italy", "Japan", "Kazakhstan", "Ko
sovo", "Kyrgyzstan", "Malaysia", "Malta", "Mexico", "Moldova", "Nepal", "New Zealand", "North Macedonia", "Norway", "Pakista
n", "Peru", "Philippines", "Poland", "Portugal", "Puerto Rico", "Romania", "Saudi Arabia", "Serbia", "Slovakia", "Slovenia",
"South Africa", "South Korea", "Spain", "Sri Lanka", "Sweden", "Switzerland", "Taiwan", "Tajikistan", "Thailand", "Turkey",
"USA", "Ukraine", "United Arab Emirates", "United Kingdom", "Vietnam"]
Matrix:

67×67 Matrix{Int64}:
 0  0  0   0  0  0  0  0  0   0  0   0  …  0  0  0  0  1  0   0  0  0   0  0
 0  0  0   0  0  0  0  0  0   0  0   1     0  0  0  0  0  0   0  0  0   0  0
 0  0  0   0  0  0  0  0  0   0  0   0     0  0  0  0  0  0   0  0  0   0  0
 0  0  0  15  0  0  0  0  0   0  0   0     0  1  0  0  0  0   1  0  0   0  0
 0  0  0   0  1  0  0  0  0   0  0   0     0  0  0  0  0  0   0  0  0   0  0
 0  0  0   0  0  5  0  0  0   0  0   0  …  0  0  0  0  0  0   0  0  0   0  0
 0  0  0   0  0  0  4  0  0   0  0   0     0  0  0  0  0  0   0  0  1   0  0
```

Sonuç

Karar ağacı modelinizin 3-fold çapraz doğrulama sonuçlarını incelediğimizde, her bir katmanın sınıflar arasındaki tahminleri gösteren bir karışıklık matrisi bulunmaktadır. Her bir katman için doğruluk (accuracy) ve kappa değerleri de verilmiştir. Şimdi, bu sonuçları yorumlayalım:

- Ortalama Doğruluk: 0.8874
- Fold 1 Doğruluk: 0.8942
- Fold 2 Doğruluk: 0.8819
- Fold 3 Doğruluk: 0.8860

Ortalama doğruluk, modelin genel performansını yansıtmaktadır. Ortalama doğruluk değeri %88,74'tür.Her bir katmanın doğruluk değeri biraz değişmekle birlikte genel olarak birbirine yakındır.Modelin performansı iyi hatta çok iyi bir seviyededir de denilebilir.

Random Forest

Rastgele orman, hem sınıflandırma hem de regresyon için kullanılan denetimli bir öğrenme algoritmasıdır. Ancak, esas olarak sınıflandırma problemleri için kullanılır. Rastgele orman algoritması, veri örnekleri üzerinde karar ağaçları oluşturur ve daha sonra her birinden tahmini alır ve son olarak oylama yoluyla en iyi çözümü seçer. Tek bir karar ağacından daha iyi olan bir toplu yöntemdir çünkü sonucun ortalamasını alarak fazla uyumu azaltır.

Rastgele orman aslında karar ağacıyla birlikte çalışır. Birden fazla karar ağacını oluşturur ve daha doğru ve istikrarlı bir tahmin elde etmek için onları birleştirir.

In [142]:
```
#build_forest argümanları:
# set of classification parameters and respective default values
# n_subfeatures: number of features to consider at random per split (default: -1, sqrt(# features))
# n_trees: number of trees to train (default: 10)
# partial_sampling: fraction of samples to train each tree on (default: 0.7)
# max_depth: maximum depth of the decision trees (default: no maximum)
# min_samples_leaf: the minimum number of samples each leaf needs to have (default: 5)
# min_samples_split: the minimum number of samples in needed for a split (default: 2)
# min_purity_increase: minimum purity needed for a split (default: 0.0)
# keyword rng: the random number generator or seed to use (default Random.GLOBAL_RNG)
#              multi-threaded forests must be seeded with an `Int`
```

In [143]:
```
#Random Forest
n_subfeatures=2;n_trees=10;partial_sampling=0.7;max_depth=-1;min_samples_leaf=5;min_samples_split=4;min_purity_increase=2.0
forest=build_forest(labels,features,n_subfeatures,n_trees,partial_sampling,min_samples_leaf,min_samples_split,min_purity_increase
```

Out[143]:
```
Ensemble of Decision Trees
Trees:      10
Avg Leaves: 31.3
Avg Depth:  5.0
```

In [144]:
```
accuracy=nfoldCV_forest(labels_test,features_test,21) #buradaki 21 depthten kaynaklı
```

```
Fold 1
Classes:  ["Australia", "Austria", "Belgium", "Bosnia Herzegovina", "Brazil", "Bulgaria", "Canada", "Chad", "Chile", "Chin
a", "Czech Republic", "Estonia", "France", "Germany", "Greece", "Hungary", "India", "Indonesia", "Iran", "Israel", "Italy",
"Japan", "Kazakhstan", "Malaysia", "Mongolia", "Netherlands", "New Zealand", "North Macedonia", "Norway", "Poland", "Portuga
l", "Romania", "Russia", "Slovakia", "Slovenia", "South Korea", "Spain", "Sweden", "Switzerland", "Syria", "Thailand", "Turk
ey", "USA", "United Arab Emirates", "United Kingdom", "Vietnam"]
Matrix:

46×46 Matrix{Int64}:
 0  0  0  0  0  0  0  0  0  0  0  0  …  0  0  0  0  0  0  0  0   0  0  1  0
 0  0  0  0  0  0  0  0  0  0  0  0     0  0  0  0  0  0  0  0   0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0     0  0  0  0  0  0  0  0   1  0  0  0
 0  0  0  0  0  0  0  0  0  2  0  0     0  0  0  0  0  0  0  0   0  0  0  0
 0  0  0  0  0  0  0  0  0  1  0  0     0  0  0  0  0  0  0  0   0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  …  0  0  0  0  0  0  0  0   0  0  0  0
 0  0  0  0  0  0  2  0  0  0  0  0     0  0  0  2  0  0  0  0   6  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0     0  0  0  0  0  0  0  0   0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0     0  0  0  0  0  0  0  0   0  0  0  0
 0  0  0  1  0  0  0  0  0  8  0  0     0  2  0  0  0  0  1  1   0  0  0  0
```

Sonuç

Ortalama doğruluk, modelin genel performansını yansıtmaktadır. Ortalama doğruluk değeri %41.11'dir. Her bir katmanın doğruluk değeri biraz değişmekle birlikte genel olarak birbirine yakındır.Modelin performansı ortalama bir seviyededir denilebilir.

k-En Yakın Komşuluk Algoritması

Bir veri noktasını etiketlemek için komşu noktaların etiketlerini kullanır. Veri noktası, çevresindeki k en yakın noktanın etiketlerine dayanarak sınıflandırılır.

Veri Adı: Cardiovascular Disease - Kardiyovasküler Hastalık

- Veriyi seçerken mümkün oldukça küçük veri seti ile çalışmak istedim çünkü k-NN'nin bir dezavantajı, veri seti büyüdükçe ve özellik sayısı arttıkça hesaplama maliyeti, hız vs. artmaktadır.

Verinin Amacı

Veri seti, bir kişinin diyabet durumunu belirlemek amacıyla kullanılabilir bir tıbbi veri setidir. Her bir satır, bir kişinin çeşitli özelliklerini temsil ederken, "Outcome" sütunu kişinin diyabet taşıyıp taşımadığını gösterir.

Veri Setinin Açıklanması

Pregnancies | Hamile: Hamile kalma sayısı
Glucose | Glikoz: Oral glikoz tolerans testinde 2 saatlik plazma glikoz konsantrasyonu
BloodPressure | Kan Basıncı: Diyastolik kan basıncı (mm Hg)
SkinThickness | Cilt kalınlığı: Triseps deri kıvrım kalınlığı (mm)
Insulin | İnsulin: 2 saatlik serum insülini (mu U/ml)
BMI: Vücut kitle indeksi (kg cinsinden ağırlık / (m cinsinden boy)^2)
DiabetesPedigreeFunction: Diyabet soyağacı
Age: Yaş
Outcome | Çıktı- Sonuç: 2 = Diyabet pozitif ; 1 = Diyabet negatif

Data Linki

https://www.kaggle.com/datasets/jocelyndumlao/cardiovascular-disease-dataset

In [82]:
```
using Pkg
Pkg.add("DecisionTree")
Pkg.add("CSV")
Pkg.add("DataFrames")
Pkg.add("MLBase")
Pkg.add("NearestNeighbors")
Pkg.add("DataStructures")

using DataStructures
using NearestNeighbors
using MLBase
using DecisionTree
using CSV
using DataFrames
```

```
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
Precompiling project...
  ✓ MLBase
  ✓ DataDeps
  ✓ PrettyTables
  ✓ ColorVectorSpace
  ✓ StatsFuns
  ✓ FFTW
  ✓ Colors
  ✓ MLJScientificTypes
  ✓ Widgets
  ✓ StatsModels
  ✓ Distributions
  ✓ HypothesisTests
  ✓ KernelDensity
  ✓ ColorSchemes
  ✓ GLM
```

In [91]:
```
using CSV
using DataFrames
di=CSV.read("C:/Users/Kev/Desktop/di.csv",DataFrame)
```

Out[91]: 394 rows × 9 columns (omitted printing of 2 columns)

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
	Int64	Int64	Int64	Int64	Int64	Float64	Float64
1	1	89	66	23	94	28.1	0.167
2	0	137	40	35	168	43.1	2.288
3	3	78	50	32	88	31.0	0.248
4	2	197	70	45	543	30.5	0.158
5	1	189	60	23	846	30.1	0.398
6	5	166	72	19	175	25.8	0.587

6	5	166	72	19	175	25.8	0.587
7	0	118	84	47	230	45.8	0.551
8	1	103	30	38	83	43.3	0.183
9	1	115	70	30	96	34.6	0.529
10	3	126	88	41	235	39.3	0.704

In [92]:
```
describe(di)
```

Out[92]: 9 rows × 8 columns

	variable	mean	min	median	max	nunique	nmissing	eltype
	Symbol	Float64	Real	Float64	Real	Nothing	Nothing	DataType
1	Pregnancies	3.2868	0	2.0	17			Int64
2	Glucose	122.305	0	119.0	198			Int64
3	BloodPressure	70.6548	24	70.0	110			Int64
4	SkinThickness	29.1066	7	29.0	63			Int64
5	Insulin	155.548	14	125.0	846			Int64
6	BMI	32.9886	0.0	33.2	67.1			Float64
7	DiabetesPedigreeFunction	0.525543	0.085	0.4495	2.42			Float64
8	Age	30.8147	21	27.0	81			Int64
9	Outcome	0.329949	0	0.0	1			Int64

In [94]:
```
using Lathe.preprocess: TrainTestSplit
train,test=TrainTestSplit(di,.60)
```

Out[94]:
```
(246×9 DataFrame. Omitted printing of 4 columns
│ Row │ Pregnancies │ Glucose │ BloodPressure │ SkinThickness │ Insulin │
│     │ Int64       │ Int64   │ Int64         │ Int64         │ Int64   │
├─────┼─────────────┼─────────┼───────────────┼───────────────┼─────────┤
│ 1   │ 1           │ 89      │ 66            │ 23            │ 94      │
│ 2   │ 3           │ 78      │ 50            │ 32            │ 88      │
│ 3   │ 2           │ 197     │ 70            │ 45            │ 543     │
│ 4   │ 1           │ 189     │ 60            │ 23            │ 846     │
│ 5   │ 5           │ 166     │ 72            │ 19            │ 175     │
│ 6   │ 1           │ 103     │ 30            │ 38            │ 83      │
│ 7   │ 1           │ 115     │ 70            │ 30            │ 96      │
│ 8   │ 3           │ 126     │ 88            │ 41            │ 235     │
│ 9   │ 13          │ 145     │ 82            │ 19            │ 110     │
│ 10  │ 3           │ 88      │ 58            │ 11            │ 54      │
⋮
│ 236 │ 2           │ 99      │ 60            │ 17            │ 160     │
│ 237 │ 3           │ 102     │ 44            │ 20            │ 94      │
│ 238 │ 1           │ 109     │ 58            │ 18            │ 116     │
│ 239 │ 13          │ 153     │ 88            │ 37            │ 140     │
```

feature selection (Train)

In [95]:
```
features = Array(train[:,[:Glucose, :Insulin, :Age]])
labels = Array(train[:,:Outcome])
features = float.(features)
labels1=string.(labels)
```

Out[95]:
```
246-element Vector{String}:
 "0"
 "1"
 "1"
 "1"
 "1"
 "0"
 "1"
 "0"
 "0"
 "0"
 "0"
 "0"
 "0"
 ⋮
 "0"
 "0"
 "0"
 "0"
```

feature_selection (Test)

In [96]:
```
features_test = Array(train[:,[:Glucose, :Insulin, :Age]])
labels_test = Array(train[:,:Outcome])
features_test = float.(features_test)
labels_test1=string.(labels_test)
```

Out[96]:
```
246-element Vector{String}:
 "0"
 "1"
 "1"
 "1"
 "1"
 "0"
 "1"
 "0"
 "0"
 "0"
 "0"
 "0"
 "0"
 ⋮
 "0"
 "0"
 "0"
 "0"
```

In [97]:
```
kdtree=KDTree(features')
```

Out[97]:
```
KDTree{StaticArraysCore.SVector{3, Float64}, Euclidean, Float64, StaticArraysCore.SVector{3, Float64}}
  Number of points: 246
  Dimensions: 3
  Metric: Euclidean(0.0)
  Reordered: true
```

In [98]:
```
?knn
```

search: **knn** **Known** **Unknown** backend_name

Out[98]:

\begin{verbatim} knn(tree::NNTree, points, k [, sortres=false]) -> indices, distances nn(tree:NNTree, points) -> indices, distances \end{verbatim}

Performs a lookup of the \texttt{k} nearest neigbours to the \texttt{points} from the data in the \texttt{tree}. If \texttt{sortres = true} the result is sorted such that the results are in the order of increasing distance to the point. \texttt{skip} is an optional predicate to determine if a point that would be returned should be skipped based on its index.

In [99]:
```
idxs,dists=knn(kdtree, features',5, true)
#5: her bir veri noktası için beş en yakın komşu bulunacak şeklinde bir k-en yakın komşu algoritması uygulayalım.
```

Out[99]:
```
([[1, 44, 211, 38, 141], [2, 141, 154, 1, 158], [3, 242, 90, 214, 60], [4, 73, 79, 190, 3], [5, 217, 34, 133, 41], [6, 118, 12
6, 19, 92], [7, 105, 63, 39, 100], [8, 107, 193, 27, 150], [9, 23, 113, 52, 230], [10, 51, 81, 174, 124]  …  [237, 19, 138, 14
4, 206], [238, 196, 136, 146, 47], [239, 28, 145, 230, 185], [240, 53, 128, 17, 231], [241, 57, 174, 170, 149], [242, 32, 60, 2
14, 90], [243, 23, 246, 88, 113], [244, 29, 195, 14, 72], [245, 65, 98, 134, 216], [246, 100, 146, 212, 168]], [[0.0, 4.5825756
9495584, 6.6332495807108, 7.14142842854285, 7.3484692283495345], [0.0, 8.602325267042627, 12.409673645990857, 13.49073756323204
2, 14.317821063276353], [0.0, 45.5411901469428, 46.14108798023731, 50.566787519082126, 51.07837115648854], [0.0, 106.0754448494
0895, 171.5459122177228, 254.53879861427805, 303.1649715913763], [0.0, 7.3484692283495345, 13.416407864998739, 20.469489490458
72, 20.54263858417414], [0.0, 5.0, 6.164414002968976, 9.1104335791443, 9.433981132056603], [0.0, 7.0, 9.0, 9.695359714832659, 1
0.723805294763608], [0.0, 6.4031242374328485, 9.1104335791443, 16.0312195418814, 17.74823934929885], [0.0, 16.64331697709324, 2
0.71231517720798, 20.97617696340303, 23.021728866442675], [0.0, 1.0, 3.0, 5.385164807134504, 6.164414002968976]  …  [0.0, 4.898
979485566356, 7.280109889280518, 7.874007874011811, 8.366600265340756], [0.0, 5.656854249492381, 9.16515138991168, 11.916375287
812984, 12.727922061357855], [0.0, 9.219544457292887, 12.041594578792296, 13.92838827718412, 14.177446878757825], [0.0, 2.23606
797749979, 12.84523257866513, 15.811388300841896, 17.46424919657298], [0.0, 6.4031242374328485, 7.681145747868608, 9.0, 9.79795
8971132712], [0.0, 36.069377593742864, 37.16180835212409, 39.71145930332956, 44.328320518603], [0.0, 10.0, 10.099504938362077,
12.206555615733702, 12.328828005937952], [0.0, 6.164414002968976, 9.1104335791443, 10.63014581273465, 16.0312195418814], [0.0,
12.96148139681572, 17.4928556845359, 20.8806130178211, 22.561028345356956], [0.0, 7.14142842854285, 7.3484692283495345, 7.81024
9675906654, 10.04987562112089]])
```

In [100]:
```
idxs
```

Out[100]:
```
246-element Vector{Vector{Int64}}:
 [1, 44, 211, 38, 141]
 [2, 141, 154, 1, 158]
 [3, 242, 90, 214, 60]
 [4, 73, 79, 190, 3]
 [5, 217, 34, 133, 41]
 [6, 118, 126, 19, 92]
 [7, 105, 63, 39, 100]
 [8, 107, 193, 27, 150]
 [9, 23, 113, 52, 230]
 [10, 51, 81, 174, 124]
 [11, 179, 36, 103, 130]
 [12, 123, 160, 166, 126]
 [13, 30, 64, 198, 33]
 ⋮
 [235, 136, 165, 59, 246]
 [236, 226, 201, 234, 208]
 [237, 19, 138, 144, 206]
 [238, 196, 136, 146, 47]
```

In [101]:
```
dists
```

Out[101]:
```
246-element Vector{Vector{Float64}}:
 [0.0, 4.58257569495584, 6.6332495807108, 7.14142842854285, 7.3484692283495345]
 [0.0, 8.602325267042627, 12.409673645990857, 13.490737563232042, 14.317821063276353]
 [0.0, 45.5411901469428, 46.14108798023731, 50.566787519082126, 51.07837115648854]
 [0.0, 106.07544484940895, 171.5459122177228, 254.53879861427805, 303.1649715913763]
 [0.0, 7.3484692283495345, 13.416407864998739, 20.46948949045872, 20.54263858417414]
 [0.0, 5.0, 6.164414002968976, 9.1104335791443, 9.433981132056603]
 [0.0, 7.0, 9.0, 9.695359714832659, 10.723805294763608]
 [0.0, 6.4031242374328485, 9.1104335791443, 16.0312195418814, 17.74823934929885]
 [0.0, 16.64331697709324, 20.71231517720798, 20.97617696340303, 23.021728866442675]
 [0.0, 1.0, 3.0, 5.385164807134504, 6.164414002968976]
 [0.0, 29.0, 35.0, 36.08323710533743, 38.40572873934304]
 [0.0, 4.47213595499958, 6.0, 6.082762530298219, 6.782329983125268]
 [0.0, 6.324555320336759, 8.306623862918075, 10.344080432788601, 12.409673645990857]
 ⋮
 [0.0, 9.486832980505138, 11.357816691600547, 11.575836902790225, 13.038404810405298]
 [0.0, 6.324555320336759, 7.874007874011811, 8.660254037844387, 10.770329614269007]
 [0.0, 4.898979485566356, 7.280109889280518, 7.874007874011811, 8.366600265340756]
 [0.0, 5.656854249492381, 9.16515138991168, 11.916375287812984, 12.727922061357855]
```

In [102]:
```
c=labels[hcat(idxs...)]
```

Out[102]:
```
5×246 Matrix{Int64}:
 0  1  1  1  1  0  1  0  0  0  0  0  0  …  0  0  0  0  0  0  0  1  1  0  0  0
 0  0  1  0  0  0  0  0  1  0  0  0  1     0  1  1  0  0  0  0  1  1  0  0  0
 0  0  0  0  1  0  1  0  0  0  1  0  0     0  0  0  0  0  0  0  1  0  0  1  0
 1  0  1  1  1  1  0  1  1  0  1  0  0     1  0  0  0  1  0  0  1  0  0  1  0
 0  0  1  1  1  0  0  1  1  0  0  0  0     0  0  1  0  0  0  0  0  0  0  0  0
```

In [103]:
```
#hcat(idxs...)
```

In [104]:
```
possible_labels=map(i->counter(c[:,i]),1:size(c,2))
```

Out[104]:
```
246-element Vector{Accumulator{Int64, Int64}}:
 Accumulator(0 => 4, 1 => 1)
 Accumulator(0 => 4, 1 => 1)
 Accumulator(0 => 1, 1 => 4)
 Accumulator(0 => 2, 1 => 3)
 Accumulator(0 => 1, 1 => 4)
 Accumulator(0 => 4, 1 => 1)
 Accumulator(0 => 3, 1 => 2)
 Accumulator(0 => 3, 1 => 2)
 Accumulator(0 => 2, 1 => 3)
 Accumulator(0 => 5)
 Accumulator(0 => 3, 1 => 2)
 Accumulator(0 => 5)
 Accumulator(0 => 4, 1 => 1)
 ⋮
 Accumulator(0 => 4, 1 => 1)
 Accumulator(0 => 4, 1 => 1)
 Accumulator(0 => 3, 1 => 2)
 Accumulator(0 => 5)
```

In [105]:
```
predictions_NN=map(i->parse(Int,string(argmax(DataFrame(possible_labels[i])[1,:]))),1:size(c,2))
```

Out[105]:
```
246-element Vector{Int64}:
 0
 0
 1
 1
 1
 0
 0
 0
 1
 0
 0
 0
 0
 ⋮
 0
 0
 0
 0
```

In [106]:
```
?confusion_matrix
```

search: **confusion_matrix**

Out[106]: No documentation found. \texttt{confusion_matrix} is of type \texttt{ROCNums\{Int64\}}. \section{Summary}

```
\begin{verbatim}
struct ROCNums{Int64} <: Any
\end{verbatim}
```

\section{Fields}

```
\begin{verbatim}
p  :: Int64
n  :: Int64
tp :: Int64
tn :: Int64
fp :: Int64
fn :: Int64
\end{verbatim}
```

In [107]:
```
confusion_matrix=MLBase.roc(Array(labels),Array(predictions_NN))
```

Out[107]:
```
ROCNums{Int64}
  p = 74
  n = 172
  tp = 44
  tn = 157
  fp = 15
  fn = 30
```

Sonuç

- p = 74
- n = 172
- tp = 44
- tn = 157
- fp = 15
- fn = 30

Accuracy (Doğruluk): (tp + tn) / (p + n) = (44 + 157) / (74 + 172) = 201 / 246 ≈ 0.8171 Bu, modelin doğruluk oranının yaklaşık olarak %81.71 olduğunu gösterir.

Precision (Kesinlik): tp / (tp + fp) = 44 / (44 + 15) ≈ 0.7458 Bu, modelin pozitif olarak tahmin ettiği örneklerin ne kadarının gerçekten pozitif olduğunu gösterir.

Recall (Duyarlılık ya da Hassasiyet): tp / p = 44 / 74 ≈ 0.5946 Bu, gerçekten pozitif olan örneklerin model tarafından ne kadarının tespit edildiğini gösterir.

Genel bir performans ölçüsü isteniyorsa Accuracy'deki başarı oranı söylenebilir. Modelin başarı oranı %81,7'dir.

Kümeleme

K-Means, Hiyerarşik Kümeleme, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), Gaussian Mixture Model (GMM) gibi çeşitli kümeleme algoritmaları bulunmaktadır. Bu algoritmalar, farklı yaklaşımlar kullanarak veri noktalarını gruplandırır.

Veri Adı: Bank Churners(Kredi Kartı Müşterileri)

- Veri setinin seçme anmacım çok fazla kategorik ve numerik değerlerinin aynı anda içermesidir.

Veri Seti Açıklaması

- **Clientnum(int):** Müşteri numarası
- **Customer Age(int):** Müşterinin yaşı
- **Gender(nominal):** Müşterinin cinsiyeti ("F" = kadın | "M" erkek)
- **Education Level(ordinal):** Müşterinin öğrenim durumu ("uneducated" = eğitimsiz | "unknown" = bilinmiyor | "high school" = lise | "college" = üniversite | "graduate" = mezun | "post graduate" = yüksek lisans | "doctorate" = doktora)
- **Marital Status(nominal):** Müşterinin medeni durumu ("Married" = Evli | "Single" = Bekar | "Divorced" = Boşanmış | "Unknown" = Bilinmiyor)
- **Card Category(ordinal):** Müşterinin kart türü ("blue" = mavi | "silver" = gümüş | "gold" = altın)
- **Months On Book(int):** Müşterinin banka ile ilişki süresi
- **Credit Limit(int):** Müşterinin kredi limiti
- **Total Revolving Bal(int):** Toplam döner sermaye
- **Total Trans AMT(int):** 12 ay içinde toplam transfer tutarı
- **Total Trans CT(int):** 12 ay içinde toplam işlem sayısı

Data Kaynağı

https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers?resource=download

- Bazı yerlerde datanın tamamı kullanılmıştır(Dbscan) ama bazı yerlerde data çok büyük olduğundan işimize yarar sütunlar alınmış ve 288 adet örneklem çekilmiştir(K- Ortalama Kümeleme,k-Medoids Kümeleme).

K- Ortalama Kümeleme

Bu algoritma, belirli bir sayıda küme (k) belirlenerek başlar. Rastgele seçilen k merkez noktası ile her veri noktası arasındaki uzaklıklar hesaplanır ve her veri noktası en yakın merkeze atanır. Daha sonra küme merkezleri güncellenir ve bu işlem belirli bir kriter sağlanana kadar tekrarlanır.

In [6]:
```
using Pkg
Pkg.add("DataFrames")
Pkg.add("CSV")
Pkg.add("Statistics")
Pkg.add("MLDataUtils")
Pkg.add("Impute")
Pkg.add("Plots")
```

```
    Updating registry at `C:\Users\Kev\.julia\registries\General`
    Updating git-repo `https://github.com/JuliaRegistries/General.git`
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 3 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 1 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 1 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 1 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 2 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 2 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
```

In [7]:
```
using CSV
using DataFrames

# Doğru dosya yolunu belirtin
csv_dosya_yolu = "C:/Users/Kev/Desktop/bank.csv"

# CSV dosyasını DataFrame'e yükle
banka = DataFrame(CSV.File(csv_dosya_yolu))
```

Out[7]: 287 rows × 10 columns (omitted printing of 4 columns)

In [7]:
```julia
using CSV
using DataFrames

# Doğru dosya yolunu belirtin
csv_dosya_yolu = "C:/Users/Kev/Desktop/bank.csv"

# CSV dosyasını DataFrame'e yükle
banka = DataFrame(CSV.File(csv_dosya_yolu))
```

Out[7]: 287 rows × 10 columns (omitted printing of 4 columns)

	Customer_Age	Gender	Education_Level	Marital_Status	Card_Category	Months_on_book
	Int64	Int64	String	String	String	Int64
1	45	0	High School	Married	Blue	39
2	49	1	Graduate	Single	Blue	44
3	51	0	Graduate	Married	Blue	36
4	40	1	High School	Unknown	Blue	34
5	40	0	Uneducated	Married	Blue	21
6	44	0	Graduate	Married	Blue	36
7	51	0	Unknown	Married	Gold	46
8	32	0	High School	Unknown	Silver	27
9	37	0	Uneducated	Single	Blue	36
10	48	0	Graduate	Single	Blue	36

In [ ]:
```julia
#banka.Credit_Limit = parse.(Float64, replace.(banka.Credit_Limit, "," => "."))
```

In [8]:
```julia
names(banka)
```

Out[8]:
```
10-element Vector{Symbol}:
 :Customer_Age
 :Gender
 :Education_Level
 :Marital_Status
 :Card_Category
 :Months_on_book
 :Credit_Limit
 :Total_Revolving_Bal
 :Total_Trans_Amt
 :Total_Trans_Ct
```

In [9]:
```julia
describe(banka)
```

Out[9]: 10 rows × 8 columns

	variable	mean	min	median	max	nunique	nmissing	eltype
	Symbol	Union…	Any	Union…	Any	Union…	Nothing	DataType
1	Customer_Age	49.7143	32	49.0	73			Int64
2	Gender	0.362369	0	0.0	1			Int64
3	Education_Level		College		Unknown	7		String
4	Marital_Status		Divorced		Unknown	4		String
5	Card_Category		Blue		Silver	3		String
6	Months_on_book	38.6132	20	36.0	56			Int64
7	Credit_Limit	10758.5	1438	6363.0	34516			Int64
8	Total_Revolving_Bal	1336.71	0	1490.0	2517			Int64
9	Total_Trans_Amt	1332.19	510	1316.0	2560			Int64
10	Total_Trans_Ct	28.5157	10	28.0	57			Int64

In [10]:
```julia
using Plots
scatter(banka.Credit_Limit, banka.Months_on_book ,color=:lightrainbow, legend=false)
```

Out[10]:

In [11]:
```julia
using Plots
scatter(banka.Total_Trans_Amt, banka.Total_Trans_Ct ,color=:lightrainbow, legend=false)
```

Out[11]:

In [12]:
```julia
?kmeans
```

```
search: PKGMODE_MANIFEST

Couldn't find kmeans
Perhaps you meant keys, ans, time_ns or kron
```

Out[12]: No documentation found. Binding \texttt{kmeans} does not exist.

In [13]:
```julia
using Pkg
Pkg.add("Clustering")
```
```
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 2 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
```

In [14]:
```julia
using Clustering
features = collect(Matrix(banka[:, 6:10])'); # features to use for clustering
k=4 #4 ve5te tekrar yapmaya başlamış
result = kmeans(features, k, display=:iter) # run K-means for the 3 clusters
```
```
  Iters               objv        objv-change | affected
-------------------------------------------------------------
      0       1.926977e+09
      1       1.357129e+09       -5.698482e+08 |        4
      2       1.341173e+09       -1.595604e+07 |        0
      3       1.341173e+09        0.000000e+00 |        0
K-means converged with 3 iterations (objv = 1.3411732751530013e9)
```

Out[14]:
```
KmeansResult{Matrix{Float64}, Float64, Int64}([36.5 39.05714285714286 39.21854304635762 36.97222222222222; 33388.5 11154.514285
714286 3509.609271523179 21535.416666666668; … ; 1269.9333333333334 1317.8142857142857 1323.58940397351 1448.111111111111; 28.7
33333333333334 27.67142857142857 28.417218543046356 30.38888888888889], [2, 2, 3, 3, 3, 3, 1, 1, 4, 2  …  3, 4, 3, 1, 4, 3, 3,
3, 3, 4], [2.770801880408168e6, 8.682118823265284e6, 1.97085560751722e6, 1.585888084338408e6, 3.358308322749004e6, 307187.36910
66131, 2.140603802222252e6, 1.8630452802222252e7, 1.800605956018567e6, 347281.8232653141  …  1.7220769585105926e6, 1.5987291622
685194e7, 550588.1174509823, 1.375670868888855e6, 2.704193344907403e6, 4.517594766457617e6, 8.49614117043113e6, 2.0720825876496
695e6, 3.7368175280470178e6, 3.3577576122685194e7], [30, 70, 151, 36], [30, 70, 151, 36], 1.3411732751530013e9, 3, true)
```

In [15]:
```julia
result.centers
```

Out[15]:
```
5×4 Matrix{Float64}:
    36.5        39.0571    39.2185     36.9722
 33388.5     11154.5     3509.61    21535.4
  1333.6      1393.11    1282.56     1456.75
  1269.93     1317.81    1323.59     1448.11
    28.7333     27.6714    28.4172     30.3889
```

In [16]:
```julia
result.counts
```

Out[16]:
```
4-element Vector{Int64}:
  30
  70
 151
  36
```

In [17]:
```julia
result.assignments
```

Out[17]:
```
287-element Vector{Int64}:
 2
 2
 3
 3
 3
 3
 1
 1
 4
 2
 3
 2
 2
 ⋮
 3
 3
 3
 4
```

In [18]:
```julia
# plot with the point color mapped to the assigned cluster index
scatter(banka.Total_Trans_Ct, banka.Total_Trans_Amt, marker_z=result.assignments,
        color=:lightrainbow, legend=false)
```

Out[18]:

Kümeleme güzel olmamış.Renkler iç içe.

In [19]:
```julia
# plot with the point color mapped to the assigned cluster index
scatter(banka.Credit_Limit, banka.Total_Trans_Amt, marker_z=result.assignments,
        color=:lightrainbow, legend=false)
```

Out[19]:

Daha düzgün bir kümeleme elde ettik. (Credit Limit ile Total_Trans_Amt arasında)

In [20]:
```julia
# plot with the point color mapped to the assigned cluster index
scatter(banka.Credit_Limit, banka.Total_Revolving_Bal, marker_z=result.assignments,
        color=:lightrainbow, legend=false)
```

Out[20]:

Daha düzgün bir kümeleme elde ettik. (Credit Limit ile Total_Revolving_Bal arasında)

In [22]:
```
names(banka)
```

Out[22]:
```
10-element Vector{Symbol}:
 :Customer_Age
 :Gender
 :Education_Level
 :Marital_Status
 :Card_Category
 :Months_on_book
 :Credit_Limit
 :Total_Revolving_Bal
 :Total_Trans_Amt
 :Total_Trans_Ct
```

In [23]:
```
features = collect(Matrix(banka[:, 6:10])'); # features to use for clustering
result = kmeans(features, 3, display=:iter)
```

```
  Iters               objv        objv-change | affected
-------------------------------------------------------------
      0       4.328191e+09
      1       3.052552e+09      -1.275639e+09 |        2
      2       2.961698e+09      -9.085396e+07 |        2
      3       2.947587e+09      -1.411034e+07 |        2
      4       2.945531e+09      -2.056778e+06 |        2
      5       2.945007e+09      -5.239038e+05 |        0
      6       2.945007e+09       0.000000e+00 |        0
K-means converged with 6 iterations (objv = 2.9450068071666474e9)
```

Out[23]:
```
KmeansResult{Matrix{Float64}, Float64, Int64}([38.7972972972973 36.577777777777776 39.07738095238095; 14205.608108108108 30368.
955555555556 3987.375; … ; 1379.972972972973 1309.7555555555555 1317.154761904762; 28.68918918918919 28.733333333333334 28.3809
5238095238], [1, 3, 3, 3, 3, 3, 2, 2, 2, 1  …  3, 1, 3, 2, 2, 3, 3, 3, 3, 2], [2.729071209641993e6, 1.8417600470025524e7, 2.355
6534938350394e6, 1.941808529549323e6, 2.4890397200255096e6, 56579.97002551705, 1.8019881207901478e7, 1.7125373190124035e6, 6.56
1722098567939e7, 6.58503112856096e6  …  3.161857386692181e6, 1.340433610153395e7, 120426.73193027824, 1.727513527456808e7, 5.47
1665827456784e7, 2.775570660501711e6, 6.311352541454092e6, 2.826326410501711e6, 2.2031900414540917e6, 1.245604263012361 5e7], [7
4, 45, 168], [74, 45, 168], 2.9450068071666474e9, 6, true)
```

In [24]:
```
# plot with the point color mapped to the assigned cluster index
scatter(banka.Credit_Limit, banka.Total_Trans_Amt, marker_z=result.assignments,
        color=:lightrainbow, legend=false)
```

Out[24]:

k-Medoids Kümeleme

k-Medoids, k-Means kümeleme algoritmasından türetilmiştir, ancak temel fark, küme merkezlerinin veri noktaları arasından seçilmesidir.Temel olarak, k-Medoids, belirli bir sayıdaki k küme merkezini seçer ve bu merkezler etrafındaki veri noktalarını bu kümelere atar. Küme merkezi, o küme içindeki diğer veri noktalarına olan benzerlik açısından en iyi temsil eden veri noktasıdır. Bu, k-Medoids'un outliers'a (aykırı veri noktalarına) daha dayanıklı olmasını sağlar.Bu algoritma, özellikle kümeleme uygulamalarında, örnek sayıları ve aykırı veri noktalarını dikkate alarak daha stabil sonuçlar elde etmek isteyen durumlar için kullanışlıdır.

In [25]:
```
Pkg.add("Distances")
using Distances
```

```
   Resolving package versions...
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Project.toml`
  No Changes to `C:\Users\Kev\.julia\environments\v1.6\Manifest.toml`
  0 dependencies successfully precompiled in 3 seconds (162 already precompiled, 6 skipped during auto due to previous errors)
  1 dependency errored. To see a full report either run `import Pkg; Pkg.precompile()` or load the package
```

In [26]:
```
scatter(banka.Credit_Limit, banka.Total_Trans_Amt,color=:lightrainbow, legend=false)
```

Out[26]:

In [27]:
```
c=Matrix(features)
```

Out[27]:
```
5×287 Matrix{Int64}:
    39    44    36    34    21    36     46  …    44    44    52    48     36
 12691  8256  3418  3313  4716  4010  34516     5556  6094  2939  5362  27126
   777   864     0  2517     0  1247   2264     1711     0  1999  1274      0
  1144  1291  1887  1171   816  1088   1330     1706   909  2434  1876   1000
    42    33    20    20    28    24     31       21    14    33    41     25
```

In [28]:
```
D=pairwise(Euclidean(),c,c,dims=2)
```

Out[28]:
```
287×287 Matrix{Float64}:
     0.0     4438.3     9335.14    9538.12  …   9912.57    7382.22   14456.6
  4438.3        0.0     4950.57    5213.48      5555.65    2980.88   18892.0
  9335.14    4950.57       0.0     2618.96      2127.22    2324.42   23724.6
  9538.12    5213.48    2618.96       0.0       1415.58    2498.22   23946.3
  8019.5     3674.82    1682.89    2903.44      3126.12    1779.01   22410.8
  8693.91    4268.09    1594.96    1451.07  …   1877.39    1565.25   23149.8
 21876.4    26297.3    31185.3    31204.4      31597.4    29175.9     7736.07
 16406.4    20833.3    25703.3    25795.0      26164.3    23721.7     2461.81
  9818.62  14192.7    19108.1    19039.8      19450.1    17043.5     5408.22
  1403.4     3499.08    8418.79    8389.53      8779.3     6321.89   15566.9
  5983.2     1626.66    3702.98    3592.09  …   4038.85    1553.76   20431.7
  3690.08    1107.85    5922.46    5858.1       6270.61    3788.06   18103.4
  1282.18    3608.79    8340.26    8813.09      9080.14    6523.55   15384.4
     ⋮                                      ⋱                ⋮
 10329.0     5916.11    2143.47    1173.52  …   1346.95    3064.97   24780.2
  9331.05    4961.55    1143.13    2554.48      2658.2     2598.21   23726.4
 10444.1     6006.43    1558.81    1893.53      1455.45    3124.58   24894.0
  5308.25    9574.57   14478.8    14380.6      14766.9    12379.7     9792.99
  8491.17    4059.55    1543.71    1675.15      2019.36    1379.43   22946.8
 21839.3    26267.9    31132.9    31224.2   …  31593.6    29156.0     7550.92
 10425.2    14820.5    19724.1    19710.3      20103.2    17683.5     4668.0
  7217.82    2860.03    2744.34    2442.75      2731.63     507.859  21649.3
```

```
 8491.17   4059.55   1543.71   1675.15      2019.36   1379.43  22946.8
21839.3   26267.9   31132.9   31224.2   …  31593.6   29156.0    7550.92
10425.2   14820.5   19724.1   19710.3     20103.2   17683.5    4668.0
 7217.82   2860.03   2744.34   2442.75      2731.63    507.859 21649.3
 6646.82   2359.45   2849.13   3760.06      4034.36   1759.19  21032.2
 9912.57   5555.65   2127.22   1415.58         0.0    2589.98  24311.8
 7382.22   2980.88   2324.42   2498.22   …   2589.98      0.0   21818.9
14456.6   18892.0   23724.6   23946.3     24311.8   21818.9       0.0
```

In [29]:
```julia
C=kmedoids(D,4, display=:iter) #tekrar yakalanamadı.
```

```
  Iters               objv        objv-change
-----------------------------------------------------
      0       6.821122e+05
      1       5.391550e+05      -1.429572e+05
      2       5.259910e+05      -1.316395e+04
      3       5.233111e+05      -2.679894e+03
      4       5.233111e+05       0.000000e+00
K-medoids converged with 4 iterations (objv = 523311.14853387483)
```

Out[29]:
```
KmedoidsResult{Float64}([251, 206, 205, 240], [2, 2, 1, 1, 1, 1, 3, 3, 4, 2  …  1, 4, 1, 3, 4, 1, 1, 1, 1, 4], [2321.9827734072
446, 2369.6373140208607, 1589.411840902162, 1050.125706760862, 2166.578870016044, 854.4553821002007, 796.1237341016785, 5439.45
8980450169, 1215.2238476922678, 1165.5655279734383  …  1116.1567990206395, 4009.302557802292, 1066.7881701631304, 246.653603257
68606, 1560.2253042429481, 2359.328506164413, 3253.6419594048757, 1231.6898960371477, 2201.4245387930064, 5787.539632693672],
[148, 73, 29, 37], 523311.14853387483, 4, true)
```

In [30]:
```julia
K=kmedoids(D,3, display=:iter) #4 ile 5 aynı değeri yakaladım. demekki k =3 olmalıymış.
```

```
  Iters               objv        objv-change
-----------------------------------------------------
      0       1.640990e+06
      1       1.146237e+06      -4.947528e+05
      2       9.638793e+05      -1.823580e+05
      3       8.392219e+05      -1.246575e+05
      4       7.883045e+05      -5.091735e+04
      5       7.688286e+05      -1.947586e+04
      6       7.522815e+05      -1.654717e+04
      7       7.522815e+05       0.000000e+00
K-medoids converged with 7 iterations (objv = 752281.4722121977)
```

Out[30]:
```
KmedoidsResult{Float64}([8, 10, 251], [2, 2, 3, 3, 3, 3, 1, 1, 1, 2  …  3, 2, 3, 1, 1, 3, 3, 3, 3, 1], [1403.4040758099572, 349
9.0761637895225, 1589.411840902162, 1050.125706760862, 2166.578870016044, 854.4553821002007, 5507.839776173595, 0.0, 6824.34253
2434901, 0.0  …  1116.1567990206395, 6087.098241362628, 1066.7881701631304, 5435.313606407638, 6112.430204100493, 2359.32850616
4413, 3253.6419594048757, 1231.6898960371477, 2201.4245387930064, 2461.80563814449], [53, 83, 151], 752281.472121977, 7, true)
```

In [31]:
```julia
K.assignments
```

```
3
3
3
3
1
1
1
2
3
2
2
⋮
3
3
3
2
3
1
1
```

In [32]:
```julia
scatter(banka.Credit_Limit, banka.Total_Trans_Amt, marker_z=K.assignments,color=:lightrainbow, legend=false)
```

Out[32]:

Biraz mor ve yeşilin karıştığını gözlemleriz ama genel olarak 3 küme yeterli gözükmektedir.

Dbscan

Yoğunluğa dayalı bir kümeleme algoritmasıdır. Yoğun veri bölgelerini kümeleyerek, düşük yoğunluktaki veri noktalarını noise olarak tanımlar. Genelde S gibi ya da yuvarlak şeklinde kümelenmiş veriler için kullanılır.

In [34]:
```
?dbscan
```

search: **dbscan** **Dbscan**Result **Dbscan**Cluster

Out[34]:

```
\begin{verbatim}
dbscan(D::DenseMatrix, eps::Real, minpts::Int) -> DbscanResult
\end{verbatim}
```

Perform DBSCAN algorithm using the distance matrix \texttt{D}. \section{Arguments} The following options control which points would be considered \emph{density reachable}:

```
\begin{itemize}
\item \texttt{eps::Real}: the radius of a point neighborhood

\item \texttt{minpts::Int}: the minimum number of neighboring points (including itself)  to qualify a point as a density point.

\end{itemize}
```

\rule{\textwidth}{1pt}

```
\begin{verbatim}
dbscan(points::AbstractMatrix, radius::Real,
      [leafsize], [min_neighbors], [min_cluster_size]) -> Vector{DbscanCluster}
\end{verbatim}
```

Cluster \texttt{points} using the DBSCAN (density-based spatial clustering of applications with noise) algorithm. \section{Arguments}

```
\begin{itemize}
\item \texttt{points}: the $d×n$ matrix of points. \texttt{points[:, j]} is a $d$-dimensional coordinates of $j$-th point

\item \texttt{radius::Real}: query radius

\end{itemize}
```

Optional keyword arguments to control the algorithm:

```
\begin{itemize}
\item \texttt{leafsize::Int} (defaults to 20): the number of points binned in each leaf node in the \texttt{KDTree}

\item \texttt{min\_neighbors::Int} (defaults to 1): the minimum number of a \emph{core} point neighbors

\item \texttt{min\_cluster\_size::Int} (defaults to 1): the minimum number of points in a valid cluster

\end{itemize}
```

\section{Example}

```
\begin{verbatim}
points = randn(3, 10000)
# DBSCAN clustering, clusters with less than 20 points will be discarded:
clusters = dbscan(points, 0.05, min_neighbors = 3, min_cluster_size = 20)
\end{verbatim}
```


```
\end{itemize}
```

Optional keyword arguments to control the algorithm:

```
\begin{itemize}
\item \texttt{leafsize::Int} (defaults to 20): the number of points binned in each leaf node in the \texttt{KDTree}

\item \texttt{min\_neighbors::Int} (defaults to 1): the minimum number of a \emph{core} point neighbors

\item \texttt{min\_cluster\_size::Int} (defaults to 1): the minimum number of points in a valid cluster

\end{itemize}
```

\section{Example}

```
\begin{verbatim}
points = randn(3, 10000)
# DBSCAN clustering, clusters with less than 20 points will be discarded:
clusters = dbscan(points, 0.05, min_neighbors = 3, min_cluster_size = 20)
\end{verbatim}
```


In [35]:
```
dbanka=pairwise(Euclidean(),features, dims=2)
```

Out[35]:
```
287×287 Matrix{Float64}:
     0.0     4438.3    9335.14   9538.12  …   9912.57   7382.22  14456.6
  4438.3        0.0    4950.57   5213.48      5555.65   2980.88  18892.0
  9335.14    4950.57      0.0    2618.96      2127.22   2324.42  23724.6
  9538.12    5213.48   2618.96      0.0       1415.58   2498.22  23946.3
  8019.5     3674.82   1682.89   2903.44      3126.12   1779.01  22410.8
  8693.91    4268.09   1594.96   1451.07  …   1877.39   1565.25  23149.8
 21876.4    26297.3   31185.3   31204.4      31597.4   29175.9    7736.07
 16406.4    20833.3   25703.3   25795.0      26164.3   23721.7    2461.81
  9818.62   14192.7   19108.1   19039.8      19450.1   17043.5    5408.22
  1403.4     3499.08   8418.79   8389.53      8779.3    6321.89  15566.9
  5983.2     1626.66   3702.98   3592.09  …   4038.85   1553.76  20431.7
  3690.08    1107.85   5922.46   5858.1       6270.61   3788.06  18103.4
  1282.18    3608.79   8340.26   8813.09      9080.14   6523.55  15384.4
     ⋮                                    ⋱               ⋮
 10329.0     5916.11   2143.47   1173.52  …   1346.95   3064.97  24780.2
  9331.05    4961.55   1143.13   2554.48      2658.2    2598.21  23726.4
 10444.1     6006.43   1558.81   1893.53      1455.45   3124.58  24894.0
  5308.25    9574.57  14478.8   14380.6      14766.9   12379.7    9792.99
  8491.17    4059.55   1543.71   1675.15      2019.36   1379.43  22946.8
 21839.3    26267.9   31132.9   31224.2   …  31593.6   29156.0    7550.92
 10425.2    14820.5   19724.1   19710.3      20103.2   17683.5    4668.0
  7217.82    2860.03   2744.34   2442.75      2731.63    507.859 21649.3
  6646.82    2359.45   2849.13   3760.06      4034.36   1759.19  21032.2
  9912.57    5555.65   2127.22   1415.58         0.0    2589.98  24311.8
  7382.22    2980.88   2324.42   2498.22  …   2589.98      0.0   21818.9
 14456.6    18892.0   23724.6   23946.3      24311.8   21818.9       0.0
```

In [36]:
```
L=dbscan(dbanka, 1000, 10)
```
Out[36]:
```
DbscanResult([2, 7], [0, 1, 1, 1, 1, 1, 2, 0, 0, 1  …  1, 0, 1, 2, 0, 1, 1, 1, 1, 0], [195, 20])
```
In [37]:
```
@show length(unique(L.assignments))
```
```
length(unique(L.assignments)) = 3
```
Out[37]:
```
3
```
In [38]:
```
scatter(banka.Credit_Limit, banka.Total_Revolving_Bal,color=:lightrainbow,  marker_z=L.assignments,legend=false)
```
Out[38]:

SON