**-Veri Ön İşleme Notları-**

* **Veri Temizleme(Data Cleaning)**
* **Gürültülü Veri(Noisy Data):**Erkek hamile olması.
* **Eksik Veri Analizi(Missing Data Analysis):** Eksik veri
* **Aykırı Gözlem Analizi(Outlier Analysis):**1 milyona kalem
* **Veri Standardizasyonu(Data Standardization)**
* **0-1 Dönüşümü(Normalization)**
* **Z-skoruna Dönüştürme(Standardization)**
* **Logaritmik Dönüşüm(Log Transformation)**
* **Veri İndirgeme(Data Reduction)**
* **Gözlem(Observation) Sayısının Azaltılması**
* **Değişken(Veriable)Sayısının Azaltılması**
* **Değişken Dönüşümleri(Veriable Transformation)**
* **Sürekli Değişkenlerde Dönüşümler**
* **Kategorik Değişkenlerde Dönüşümler**

Aykırı Değer Yakalama

import seaborn as sns

df = sns.load\_dataset('diamonds')

df = df.select\_dtypes(include = ['float64', 'int64'])

df = df.dropna()

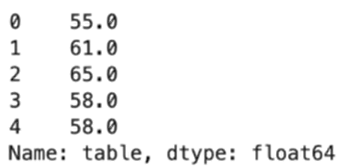
df.head()

* 3. Satırda çektiğimiz değişken içerisinden sadece sayısal değişkenleri aldık.
* 5. Satırda eksik değerleri almayarak devam ediyoruz.

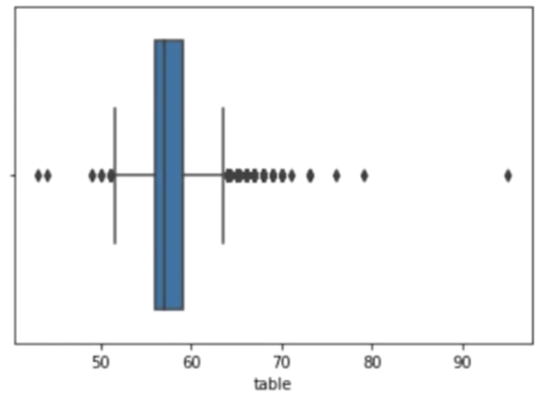


df\_table = df["table"]

df\_table.head()



sns.boxplot(x = df\_table);



Q1 = df\_table.quantile(0.25)

Q3 = df\_table.quantile(0.75)

IQR = Q3-Q1

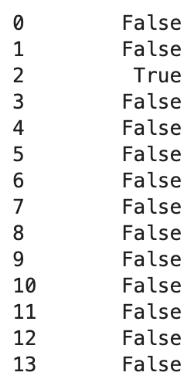
* Q1 = 56.0 , Q3=59.0, IQR=3.0
* Q1 yüzde 25 lik , Q3 yüzde 75 lik kısmı ifade ediyor.
* Quantile argümanı küçükten büyüğe sıralar.

alt\_sinir = Q1- 1.5\*IQR

ust\_sinir = Q3 + 1.5\*IQR

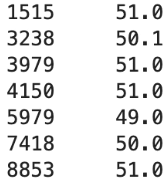
* Alt Sınır ve Üst sınır değerlerimizi ayarladık.
* Altsınır:51,5 Üstsınır:63,5

(df\_table < alt\_sinir) | (df\_table > ust\_sinir)



* Alt sınırdan küçük olanları veya üst sınırdan büyük olanları(Aykırı değerleri) true gösterdi.
* Eğer bunların değerlerini görmek istersek yapmamız gerekenler aşağıdadır.

aykiri\_tf = (df\_table < alt\_sinir) | (df\_table > ust\_sinir)

df\_table[aykiri\_tf]

* Aykırı değerler indexi ve değeri kaç olduğu gösterdik bu şekilde.
* Sadece İndexlerini istersek aşağıdaki kodun girilmesi yeterlidir.

**df\_table[aykiri\_tf].index**

Aykırı Değer Problemlerini Çözme:

Silme İşlemleri:

type(df\_table)

df\_table = pd.DataFrame(df\_table)

df\_table.shape

* Tipi : Pandas core series, tipi dataframe ye çevirdik.
* Boyutu: (53940, 1)

t\_df = df\_table[~((df\_table < (alt\_sinir)) | (df\_table > (ust\_sinir))).any(axis = 1)]

* Bu kod ile t\_df isimli bir dataframe oluşturduk. Bunun içerisine aykırı değerler dışındakileri almasını sağladık.
* ~ sembolü sağındaki değerlerin dışındakilerini al.
* any(axis = 1) değeri girmemizin sebebi sadece sutun bazında işlem yapacağımızı belirttik.

t\_df.shape

* (53335, 1) değer kaldı. Yaklaşık 605 değer uçtu. :)

Ortalama Değer İle Doldurma İşlemleri:

import seaborn as sns

import pandas as pd

df = pd.read\_csv('diamonds.csv')

df = df.select\_dtypes(include = ['float64', 'int64'])

df = df.dropna()

df.head()

* Yukarıdaki işlemlerin aynısını yaptık. Bu kısımda diamonds benim bilgisayarımda hata verdiği için indirip okumasını sağladık.(3.satır)
* 4. Satırda sadece sayısal değişkenleri aldık.
* 6. Satırda eksik değerleri almadan devam ediyoruz.

df\_table = df["table"]

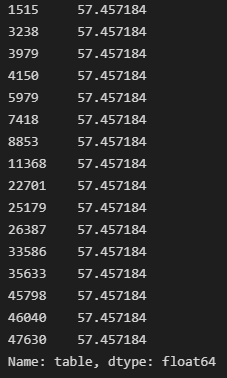
aykiri\_tf.head()

df\_table[aykiri\_tf]

df\_table.mean()

df\_table[aykiri\_tf] = df\_table.mean()

df\_table[aykiri\_tf]



* **Yukarıdaki işlemlerle aykırı değerleri yakaladık zaten. 4. Satırla sayısal değişkenlerimizin ortalamasını aldık ve 5. Satırla aykırı değerlerimize ortalamayı ekledik. Böylelikle standart sapmamız olmaz.**

Baskılama yöntemleri: Alt değerlerin altında kalan değerler alt değere eşitlenip üst değer üstündeki değerlerde üst değere eşitlenir.

df\_table = df["table"]

Q1 = df\_table.quantile(0.25)

Q3 = df\_table.quantile(0.75)

IQR = Q3-Q1

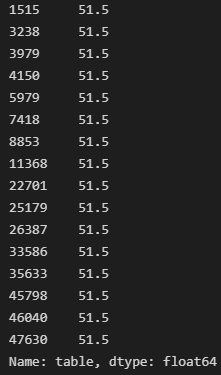
alt\_sinir = Q1- 1.5\*IQR

aykiri\_tf = (df\_table < alt\_sinir)

df\_table[aykiri\_tf] = alt\_sinir

df\_table[aykiri\_tf]

* Alt sınırımız 51.5 değeriydi. Alt sınırımızın altında kalan aykırı değerleri alt sınırımıza eşitledik. Aynısı yukarısı içinde yapılabilir.



Aykırı Değer Problemlerini Çözme:

* Şimdi 17 28 49 yaşlarında olmak aykırı durum değil. 2 3 kere evlenmekte aykırı durum değil. Lakin 17 yaşında 3 kere evlenmek aykırı durum olabiliyor.

**Local Outlier Factor(Yerel Aykırı Değer Faktörü):** Bir noktanın local yoğunluğu bu noktanın komşuları ile karşılaştırılıyor. Eğer bir nokta komşularınının yoğunluğundan anlamlı şekilde düşük ise bu nokta komşularından daha seyrek bir bölgede bulunuyordur yorumu yapılabiliyor. Dolayısıyla burada bir komşuluk yapısı söz konusu. Bir değerin çevresi yoğun değilse demek ki bu değer aykırı değerdir şeklinde değerlendiriliyor.

import seaborn as sns

import pandas as pd

df = pd.read\_csv('diamonds.csv')

diamonds = df.select\_dtypes(include = ['float64', 'int64'])

df = diamonds.copy()

df = df.dropna()

df.head()

* Her zamanki işlemler yapıldı.

import numpy as np

from sklearn.neighbors import LocalOutlierFactor

clf = LocalOutlierFactor(n\_neighbors = 20, contamination = 0.1)

* n\_neighbors : komşuluk sayısı ifade eder.
* Contaminantion : Yoğunluk ifade eder.

clf.fit\_predict(df)



* Her gözlem birimi için elimizde skorlar oldu.

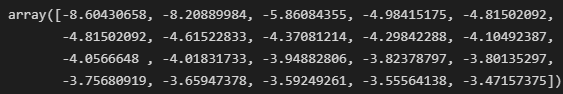
df\_scores = clf.negative\_outlier\_factor\_

df\_scores[0:10]



* Negatif skorlarımızı aldık.

np.sort(df\_scores)[0:20]



esik\_deger = np.sort(df\_scores)[13]

aykiri\_tf = df\_scores > esik\_deger

aykiri\_tf

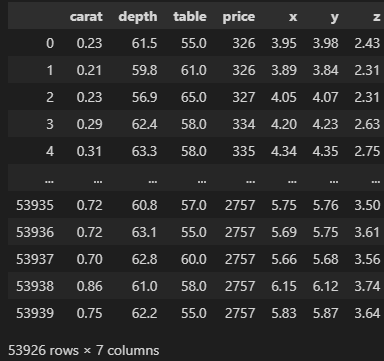


* Sort sistemiyle – değerlerimizi sıraladık.
* Daha sonra 13 değerimizi eşik değer olarak kabul ettik.
* Aykiri\_tf ile aykırı değerlerimizi seçtik.

**Silme Yöntemi:**

yeni\_df  = df[df\_scores > esik\_deger]

yeni\_df



* Şeklinde yaptığımızda yeni\_df aykırı değerlerden arındırılmış dataframe oldu.
* Eğer aykırı değerlere ulaşmak istersek’de kod aşağıdadır.

df[df\_scores < esik\_deger]

**Baskılama Yöntemi:**

df[df\_scores == esik\_deger]



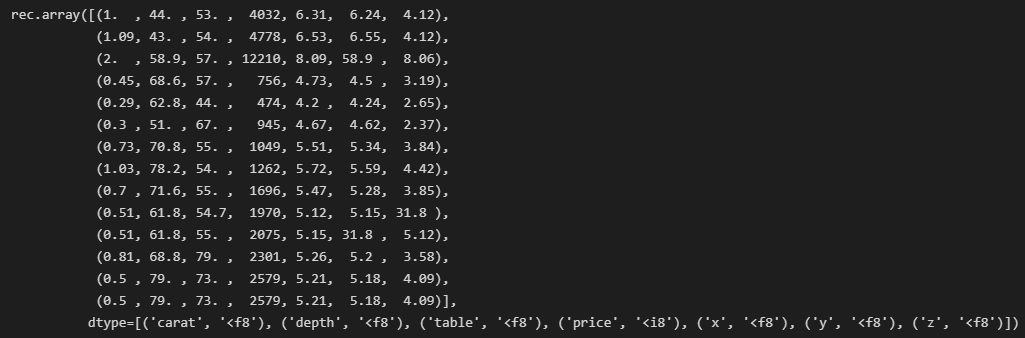
* Eşik Değerimize karşılık gelen değerler yukarıda verildiği gibidir.

baski\_deger = df[df\_scores == esik\_deger]

aykirilar = df[~aykiri\_tf]

* Şimdi elimizde bir aykırı değerlerimiz var birde eşik değerlerimiz var.
* İlk olarak aykırıları index’siz array’e çevireceğiz.
* Daha sonra baskı değeride arrayleştirip atama işlemi yapacağız

res = aykirilar.to\_records(index = False)

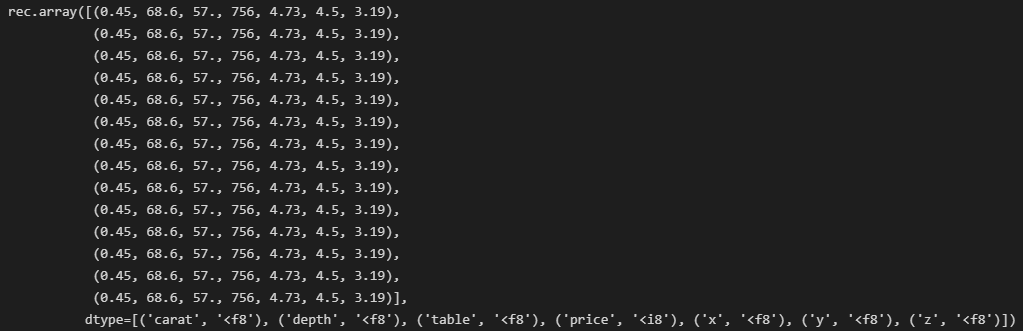


* Elimizdeki aykırı değerlerini index olmadan kaydettik.

res[:] = baski\_deger.to\_records(index = False)

* Elimizdeki res değerleri yerine baskı değerdekileri indexleri olmadan kaydettik.

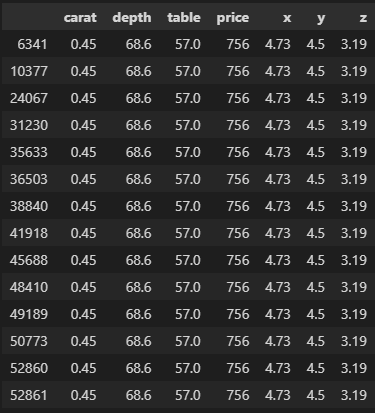
res



* Yeni res değerlerimiz bunlar oldu. (Son bir işlem kaldı.)

df[~aykiri\_tf] = pd.DataFrame(res, index = df[~aykiri\_tf].index)

df[~aykiri\_tf]



* 1. Satırda aykırı değerlerimizin yerine resdeki değerlerimizi yaz yalnız indexlerimiz dfdeki indexlerimiz olmasını sağladık.
* 3. Satırdada aykırı değerlerimizi görüntüledik.

Eksik Veri Analizi

* İncelenen veri setinde gözlemlerde eksiklik olması durumunu ifade eder.
* İlk sorulması gereken sorular bunlardır:
* Eksik veri yapısal bir sorun mu(mesela toplam kredi kartı harcaması gözükmüyorken farklı bir değişken vardı kredi kartı varmı diye onda 0 ise harcama olmaması normal)
* Rasgele herhangi bir sorundan da oluşmuş olabilir.
* NA her zaman eksik anlamına gelmez.(Kredi kartı örneğindeki gibi 0 veya ölçülemediğini gösterir.)
* Bilgi kaybıda gerçekleşebilir.(array silerken)
* Eksik Veri Türleri:
* Tümüyle Raslantısal Kayıp: Rasgele kayıp yapısaldan problemlerden kaynaklanmayan.
* Raslantısal Kayıp : Diğer d eğişkenlerle ilişkisi olan kayıp.
* Raslantısal Olmayan Kayıp: Göz ardı edilemeyecek olan ve yapısal problemlerden ortaya çıkan eksiklik türü.

import numpy as np

import pandas as pd

V1 = np.array([1,3,6,np.NaN,7,1,np.NaN,9,15])

V2 = np.array([7,np.NaN,5,8,12,np.NaN,np.NaN,2,3])

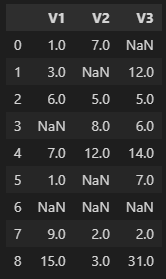
V3 = np.array([np.NaN,12,5,6,14,7,np.NaN,2,31])

df = pd.DataFrame({"V1" : V1,

         "V2" : V2,

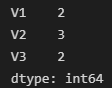
         "V3" : V3})

df



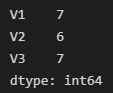
* Kendimiz bir data frame hazırlayıp listede göründüğü gibi eksik veriler NaN olarak gözükmektedir.

df.isnull().sum()



* İsnull ile eksik verileri , sum ile toplamda kaç tane olduğunu görebilirsiniz. V1’de 2 tane olduğu gibi.

df.notnull().sum()

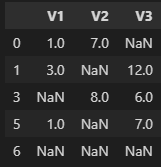


* Notnull ile sadece eksik olmayan değerlere de ulaşılabilir.

df.isnull().sum().sum()

* Veride toplam kaç tane eksik değer olduğunu gösterir.(7)

df[df.isnull().any(axis = 1)]



* df.isnull() yazdığımızda değerler yerinde True-False değerlerini gösterir. any ile en az bir tane true değeri dönderen satırları getirmesini sağladık.

df[df.notnull().all(axis = 1)]



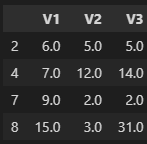
* Fonksiyonu sonuna all getirerek sadece hepsi dolu olanları getirmesini sağladık.

df[df["V1"].notnull() & df["V2"].notnull()& df["V3"].notnull()]

* Yukarıdaki ile aynı çıktıyı verecektir.

**Eksik değerlerin silinmesi:**

df.dropna()



* 1 tane dahi eksik değer olursa bütün satırı sildi. Kalıcı için silmesi için aşağıdaki kod girilmeli.

df.dropna(inplace=True)

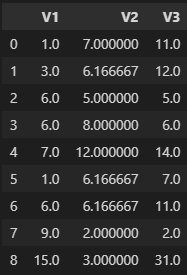
**Basit değer atama:**

df["V1"].fillna(df["V1"].mean())

df["V2"].fillna(0)

* V1 deki boş değerler yerine V1 in ortalamasını alıp doldurduk.
* V2 deki boş değerler yerine ise 0 değerini doldurduk.(fillna())

df.apply(lambda x: x.fillna(x.mean()), axis = 0)

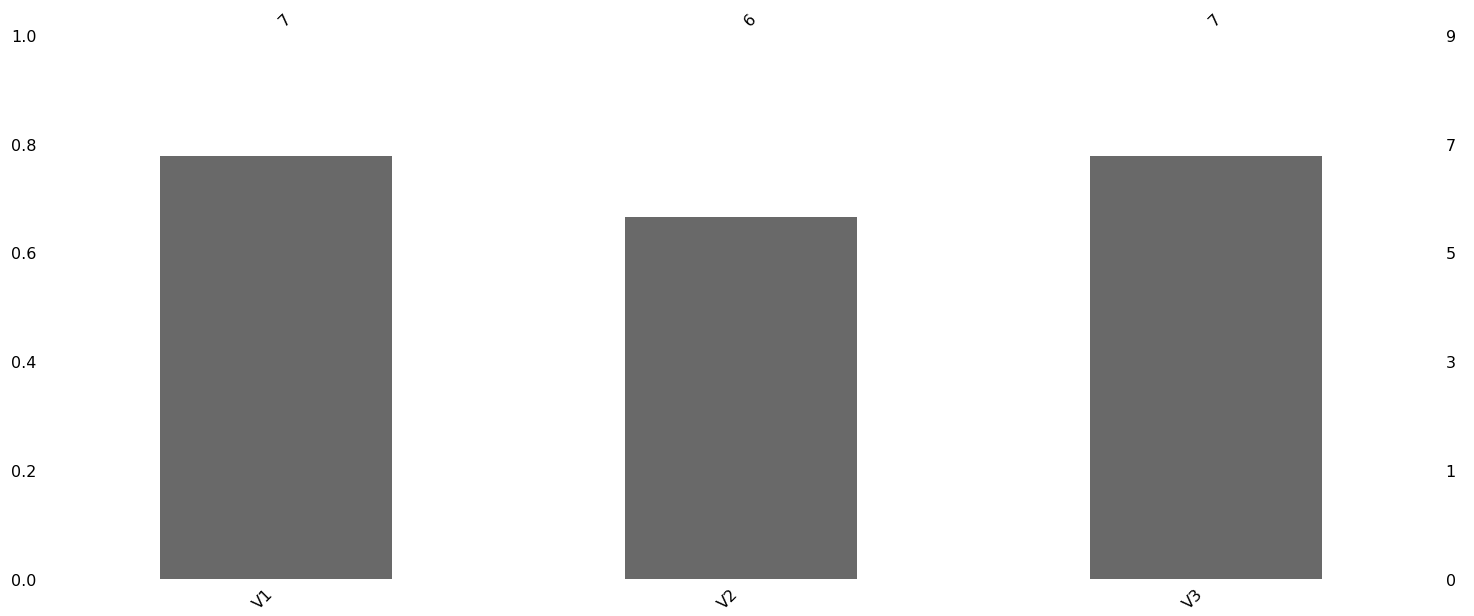


* Yukarıdaki fonksiyon ile boş olan yerlere ortalama değer ile doldurmasını sağladık.

**Veri eksikliğini görselleştirme:**

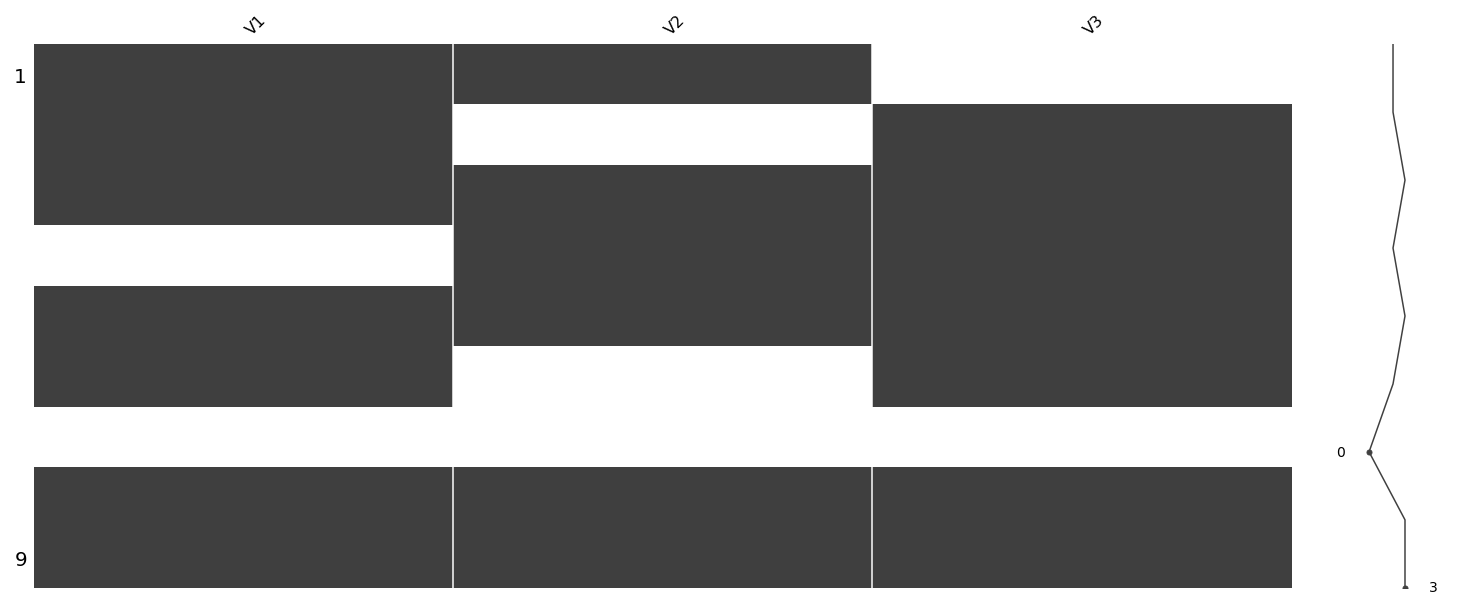
import missingno as msno

msno.bar(df);



* Cubuklar üzerindeki sayılar değişkenlerdeki toplam tam değişken sayısını ifade eder.(V1:7 , V2:6 , V3:7)
* Soldaki eksen eksikliklerin yada tam olmanın yüzdesini gösterir
* Sağ eksen ise elimizdeki verinin gözlem sayısını gösterir.

msno.matrix(df);

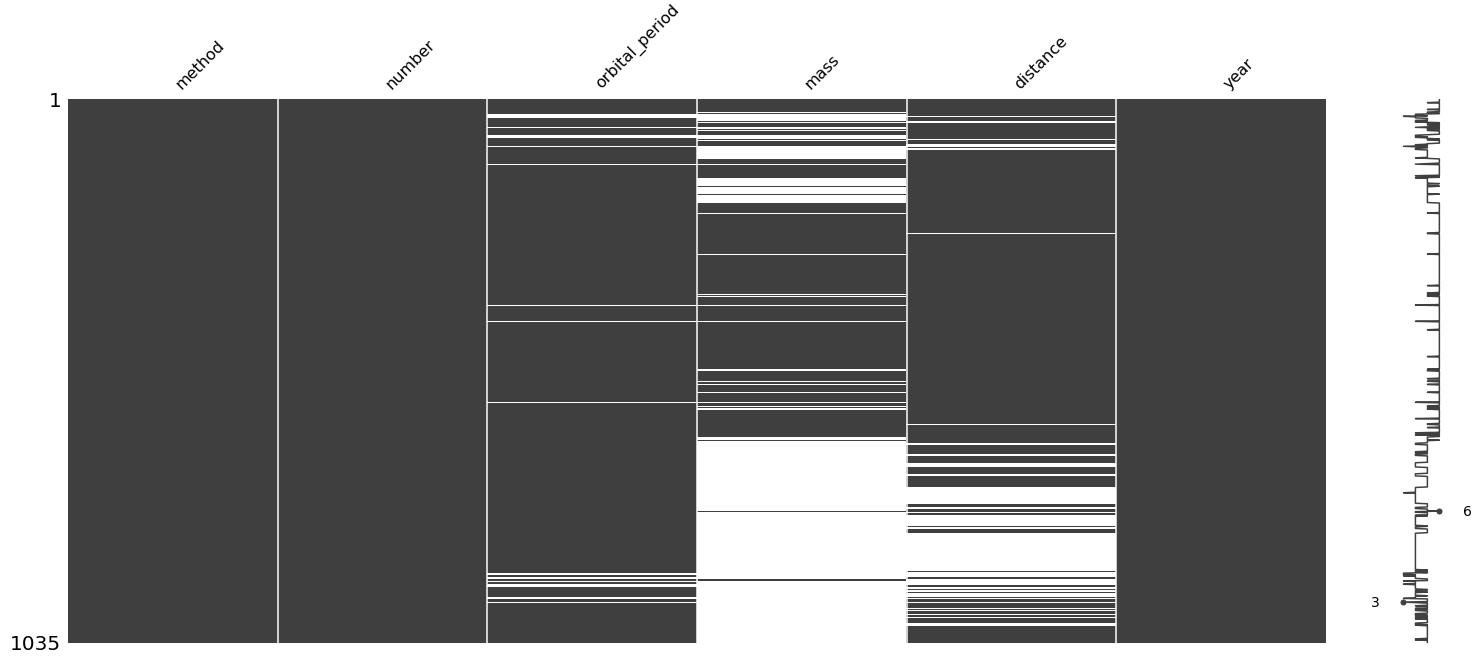


* Grafik date framemizdeki boş olan kısımları beyaz ile göstermiştir. Örnek 1. Satır en sağ boş mesela.
* Sağ taraftaki çubuk ise dolu olan kısımları gösteriyor. 3 değişkenimiz var hepsi boş ise 6. Satırda gösterildiği gibi 0’ı gösteriyor.

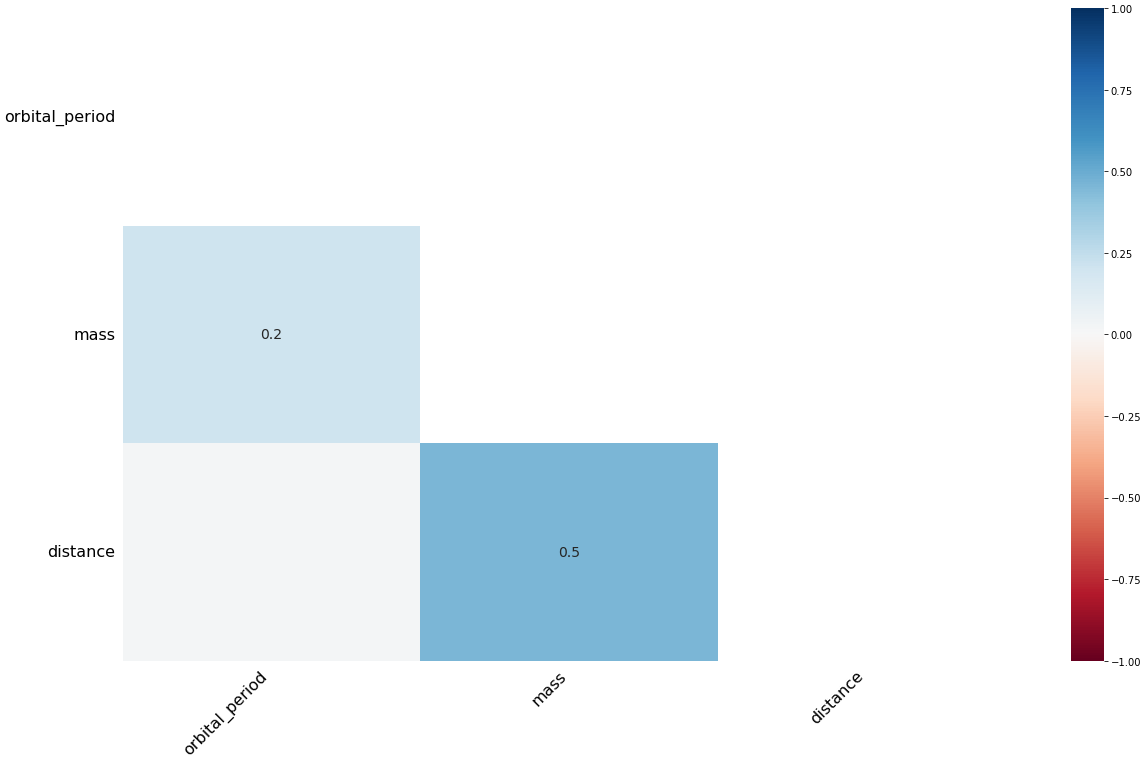
import seaborn as sns

df = sns.load\_dataset('planets')

msno.matrix(df);



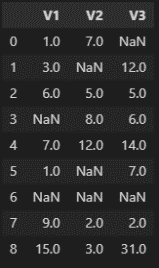
* Daha yoğun planets datasında denedik ve orbital\_period ile mass arasında bir ilişki olabilir. Orbitalde yokken massda da yüksek ihtimalle yok.(Kredi kartındaki kart olması ile harcama)



* Bu grafik bize değişkenlerin birbiri ile arasında bağ olabildiğini gösteriyor. (distance ile mass arası %50 bağ olabilir)
* Mass ile orbital arası %20 bağ olabilir)

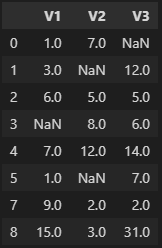
**Eksik değerlerin silme yöntemleri:**

df



* Yukarıdaki data framemizden devam edeceğiz.(3. Sayfa sonundaki)

df.dropna(how = "all")



* How ile nasıl sileceğimizi sordu. Bizde full boş olan satırı silmesini istedik.(6.satır)

df.dropna(axis = 1)



* Değişken bazında bir tane eksik değer bulunanı sil dediğimiz için değişken kalmadı. :)
* how =”all” değerini ekleseydik değişken bazında tüm değerleri NaN olan değişkenler(sutunlar) gidecekti.

**Basit değer atama yöntemleri 1 (sayısal):**

df["V1"].fillna(0)

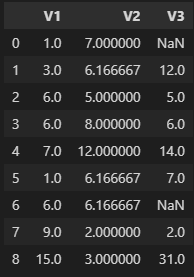
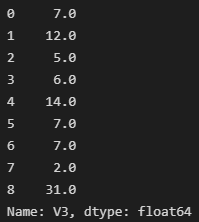
* Bu yöntemler yukarıda açıklaması vardır.

df.fillna(df.mean()[:])

* Yukarıdaki hepsini ortalama ile doldurmanın farklı yolu.
* V1 ve V2 değişkenlerinin dağılımları düzgün ise ortalama ile doldurulabilir fakat V3 ise dağılımı simetrikse(sağa-sola çarpık) yapısal bir bozukluk varsa median değeri daha iyi olur.

df.fillna(df.mean()["V1":"V2"])

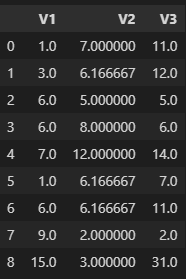
df["V3"].fillna(df["V3"].median())



* Yukarıdaki satırla v1 ve v2 nin ortalaması ile doldurup v3’u median ile doldurduk.
* Sol 1.satır sağdaki fotoğraf 2. Satırın çıktısı.

df.where(pd.notna(df), df.mean(), axis = "columns")

* 3. Yol olarakta boş olan değerleri ortalama ile doldurması için yazılan bir kod bloğu.



**Basit değer atama yöntemleri 2 (Kategorik Değişken kırılımı):**

import numpy as np

import pandas as pd

V1 = np.array([1,3,6,np.NaN,7,1,np.NaN,9,15])

V2 = np.array([7,np.NaN,5,8,12,np.NaN,np.NaN,2,3])

V3 = np.array([np.NaN,12,5,6,14,7,np.NaN,2,31])

V4 = np.array(["IT","IT","IK","IK","IK","IK","IK","IT","IT"])

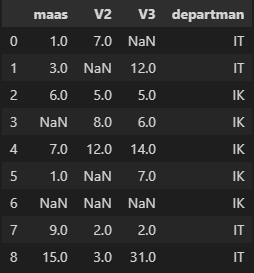
df = pd.DataFrame({"maas" : V1,

        "V2" : V2,

          "V3" : V3,

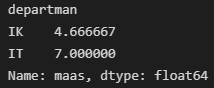
        "departman" : V4})

df

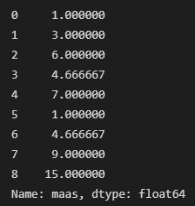


* Veri setimizde departmanlar var ve bu kişilerin maaşları var. Eksik değerleri departmanları göz önünde bulundurarak eksik değerleri doldurmayı planlıyoruz.

df.groupby("departman")["maas"].mean()



df["maas"].fillna(df.groupby("departman")["maas"].transform("mean"))



* Departmanına göre eksik verilere departmanların ortalamasını alarak doldurduk.

**Basit değer atama yöntemleri 3 (Kategorik Değişkenlerde):**

V1 = np.array([1,3,6,np.NaN,7,1,np.NaN,9,15])

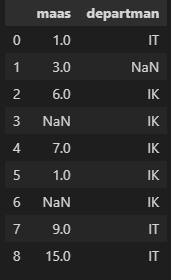
V4 = np.array(["IT",np.nan,"IK","IK","IK","IK","IK","IT","IT"], dtype=object)

df = pd.DataFrame(

        {"maas" : V1,

        "departman" : V4})

df



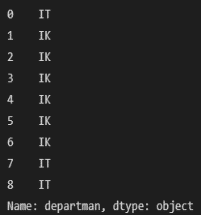
* Kategorik değişkenlerde eksiklik olduğu zaman en mantıklı yöntem mod(en çok tekrar eden değer) ile doldurmaktır.

df["departman"].mode()[0]



* Sonuna [0] değerini koyarsak yukarıdaki gibi string ifade çıkar. Dikkat edilmesi gereken noktadır.

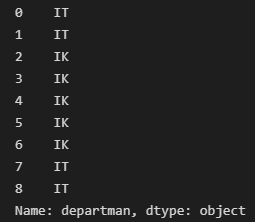
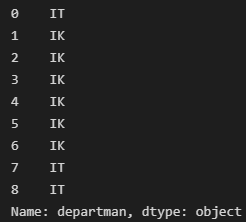
df["departman"].fillna(df["departman"].mode()[0])



* Sonuç olarak 1. Değere IK değerini atamış olduk.
* Kalıcı değildir. İnplace = True ile kalıcı hale getirilebilir.

df["departman"].fillna(method = "bfill")

df["departman"].fillna(method = "ffill")



* 1. Kod ile boş olan kısmı alttaki ile doldur. (sol)
* 2.Kod ile boş olan kısmı üstteki ile doldur. (sağ fotoğraf)

**Değer atama yöntemleri 4 (Tahmine dayalı(KNN,EM,Random)):**

import seaborn as sns

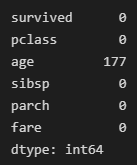
import missingno as msno

import numpy as np

df = sns.load\_dataset('titanic')

df = df.select\_dtypes(include = ['float64', 'int64'])

df.isnull().sum()



* Titanic verimizin içerisinde age değişkeni içerisinde 177 eksik gözlem olduğunu tespit ettik.
* Makine öğrenmesi ile 3 taktikle beraber doldurma işlemi gerçekleştireceğiz.

from ycimpute.imputer import knnimput

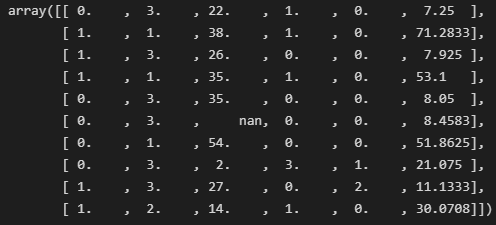
var\_names = list(df)

n\_df = np.array(df)

n\_df[0:10]

n\_df.shape

* Knn input fonksiyonu bizden 1 array bekliyor. İlk olarak isimleri saklayıp data frameyi array’e çevireceğiz. Atama işlemleri sonrası tekrar isimlendirmeleri yapacağız.
* Değişken isimlerini tutuyoruz(2.satır)
* Daha sonra çevirme işlemi gerçekleşti.(3.satır)
* 4. Ve 5. Satırın çıktıları aşağıdaki gibidir.



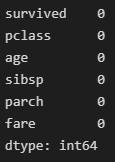
dff = knnimput.KNN(k = 4).complete(n\_df)

* K:komşuluk sayısını ifade eder.
* Complate: doldurma argümanı.(n\_df) yi doldurmasını istedik.

dff = pd.DataFrame(dff, columns = var\_names)

* Array’i data frame’ye çevirdik. Değişkenleride daha önceden sakladığımız var\_names listesinden aldı.

dff.isnull().sum()



* Kontrol edileceği üzere eksik veri kalmadığını gözlemlemiş olduk.

Random Forests Doldurması.

from ycimpute.imputer import iterforest

dff = iterforest.IterImput().complete(n\_df)

* Diğer yapılan işlemlerin hepsi aynı. Sadece knn girdiğimiz yere bu şekilde giriş yapıyoruz.

EM

from ycimpute.imputer import EM

dff = EM().complete(n\_df)

* Diğerleri ile yapılan işlemler aynı. İlk olarak değişken isimlerini sakla, sonra dataframeyi array’e cevir. Yukarıdaki kodu yaz. Daha sonra dff’i data frameye çevir.
* Makine kodları çalıştırırken hata alırsanız takılmayın. Kod çalışmış oluyor ve doldurma işlemi gerçekleşmiştir.

Veri Standardizasyonu(Değişken Standardizasyonu)

* Değişken standardizasyonu ile değişken dönüşümü arasında fark vardır.
* Değişken standartlaştığında değişkenin değeri değişecektir lakin taşımış olduğu yayılım, dağılım bilgisinin özütü mevcutta kalacaktır. Ör: a:10 ve listede 23. Sırada, standartlaştığında 2 3 gibi değerler almış olsa da büyükten küçüğe sıralandığında yine 23 kalacaktır.
* Değişken dönüştürmek ise dönüştürülen değerin taşındığı şekilde kalamamasıdır. Ör: Cinsiyet E/K den 0/1 yaptığımızda yapı değişmiş olacaktır.

import numpy as np

import pandas as pd

V1 = np.array([1,3,6,5,7])

V2 = np.array([7,7,5,8,12])

V3 = np.array([6,12,5,6,14])

df = pd.DataFrame(

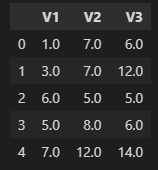
        {"V1" : V1,

         "V2" : V2,

         "V3" : V3})

df = df.astype(float)

df

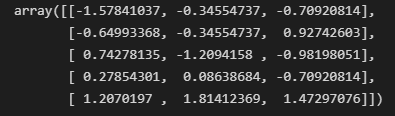


* Data frame oluşturduk. 10. Satırda hepsinin tipini ondalık sayıya çevirdik.

**Standardizasyon:** Değerleri -3 +3 aralığında dönüştürmeyi sağlıyor.

from sklearn import preprocessing

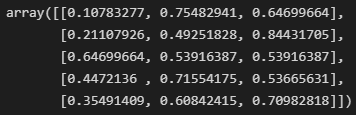
preprocessing.scale(df)



* Scale: dönüştürmek için kullanılır.
* Bu tip dönüştürme fonksiyonlarında copy: True veya inplace: False olduğu için orijinal yapı bozulmaz.

**Normalizasyon:** Değerleri 0-1 aralığında dönüştürmeyi sağlıyor.

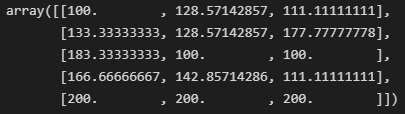
preprocessing.normalize(df)



**Min-Max Dönüşümü:** Bizim belirlediğimiz min-max aralığında dönüştürme işlemini sağlıyor.

scaler = preprocessing.MinMaxScaler(feature\_range = (100,200))

scaler.fit\_transform(df)

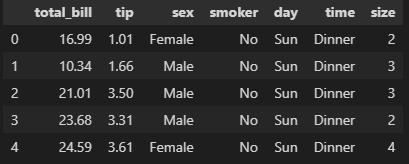


* İstenirse 4. Satır scalerden sonrası 2. Satır sonuna da bağlanabilirdi.
* Feature range ile aralığı belirleyip devamına 4.satır yazılmalı.

**Değişken Dönüşümleri:** Her bir standardizasyon değişken dönüşümü demektir. Tek fark yapısının bozulup bozulmaması.

import seaborn as sns

df = sns.load\_dataset('tips')

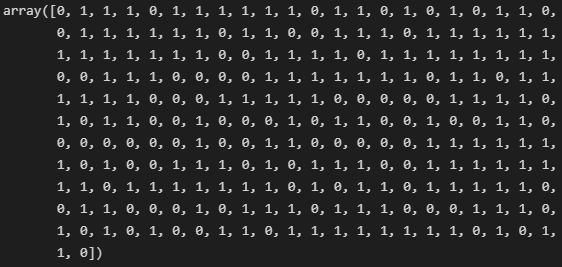
df.head()

* **0-1 Dönüşümü**
* **1 Ve Diğerleri(0) Dönüşümü**
* **Çok Sınıflı Dönüşüm**

**0-1 Dönüşümü:**

from sklearn.preprocessing import LabelEncoder

lbe = LabelEncoder()

lbe.fit\_transform(df["sex"])

* Görüldüğü üzere kadınlar 0 erkekler 1 olarak dönüştürüldü.
* Eklememizi yapalım şimdide.

df["yeni\_sex"] = lbe.fit\_transform(df["sex"])

df

* Yeni değişken’e bu 0-1 değerlerini atadık. Sex yerine 0-1 yazmamızı istersek df[“sex”] yerine yazmamız yeterli.

**1 Ve Diğerleri(0) Dönüşümü:** Bir değişkendeki 1 değeri seçip 1 değerini ver diğerlerine 0 ver. Ör:pztsi 1 Salı Çarşamba .. 0 olsun.

import numpy as np

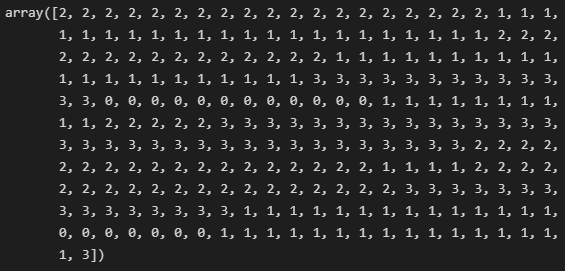
df["yeni\_day"] = np.where(df["day"].str.contains("Sun"), 1, 0)

* Where içindeki koşulu sağlayanlara 1 sağlamayanlara 0 ver.
* Contains içerdiği değer aramada kullanılır(True-False dönderir)
* Böylelikle sun olan günler yeni\_day da 1 diğerleri 0 oldu.

**Çok Sınıflı Dönüşüm:** Bir değişken içerisindeki kategorik değişkenler sayılara dönüştürülür. Ör: pazartesi 1, Salı 2, Çarşamba 3 gibi…

from sklearn.preprocessing import LabelEncoder

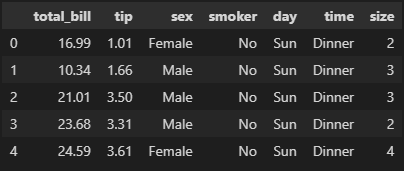
lbe = LabelEncoder()

lbe.fit\_transform(df["day"])

* Böylelikle sun,sat,thur gibi 4 değişkene sayılar verildi.
* Dikkat edilmesi gerek durum ise bu kategorikleri sayılara çevirdik lakin makine bunları sayısal işlem olarak kullanabilir. Bunun içinde aşağıdaki sistemden devam ediyoruz.

**One-Hot Dönüşümü ve Dummy Değişken Tuzağı:** Kategorik değişkenler içerisindeki sınıfların etkisini veri setinde hissedilebilir hale dönüştürmemize yardımcı oluyor.

df.head()

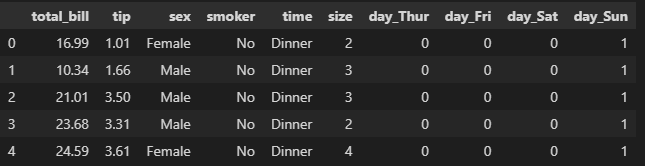


df\_one\_hot = pd.get\_dummies(df, columns = ["sex"], prefix = ["sex"])

df\_one\_hot.head()



* 2 değişken olduğunda sex\_Male ve sex\_Famale aynı şeyi ifade etmektedir. Bunun için 1 tane dummy değişkeni olması gerekir.

pd.get\_dummies(df, columns = ["day"], prefix = ["day"]).head()

* Burada bir problem yoktur çünkü 4 tane değerde farklı durumları ifade ediyor. 2 değişkenlide 0-1 olayı aynı ifadeyi söylemiş oluyor bize. Buna da Dummy tuzağı deniyor.