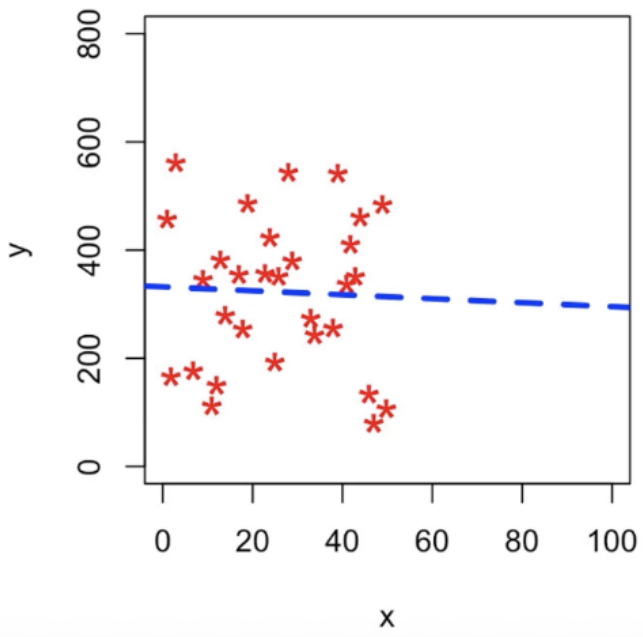
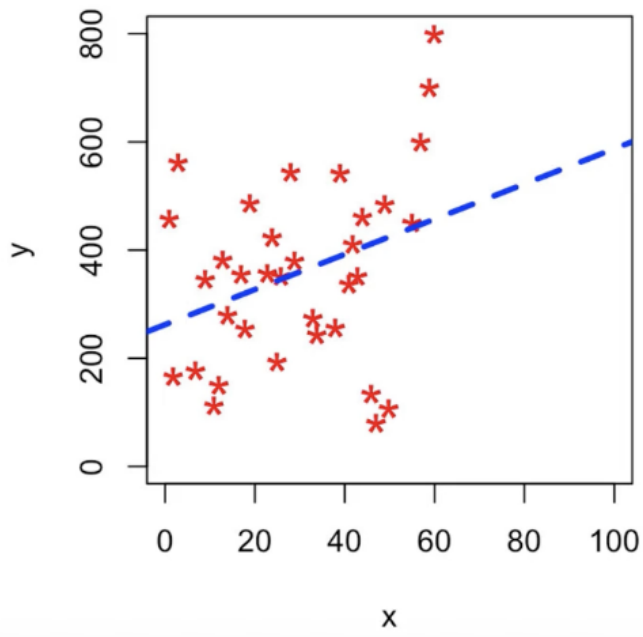
Outliers

Verideki genel eğilimin *oldukça* dışına çıkan değerlere outlier (aykırı değer) denir.

Aykırı değerler neye sebep olur ? Ne gibi problemler ortaya çıkarır ?





1. grafikte herhangi bir aykırı değer bulunmamaktadır. x ve y arasındaki ilişki, ilişkinin yönü ve şiddeti bir doğru aracılığı ile modellenmiştir.

2. grafikte bu veri setine 3 tane aykırı değer eklenmiştir. x ve y'nin ilişkisini ifade eden doğrunun yönü ve şiddeti değişime uğramıştır. x ve y arasında normalde negatif bir ilişki tanımı varken, bu ilişki pozitife dönüşmüştür.

Özellikle doğrusal problemlerde aykırı değerlerin etkileri daha şiddetlidir. Ağaç yöntemlerinde bu etkiler daha düşüktür. Yine de her türlü problemde, aykırı değerler göz önünde bulundurulması gereken değerlerdir.

Aykırı değerler neye göre belirlenirler ?

1. Sektör Bilgisi

   Örneğin bir ev fiyat tahmin modelinde 1000 m2 lik evler çalışmanın dışında bırakılabilir.

2. Standart Sapma Yaklaşımı

   Bir değişkenin ortalamasının 10, standart sapmasının 5 olduğu varsayılsın. Standart sapmanın altında ya da üstünde kalan değerler, yani 5'in altındaki ve 15'in üstündeki değerler aykırı değer olarak kabul edilir.

   Bir diğer örnekte, bir değişkenin ortalamasının 100, standart sapmasının 10 olduğu varsayılsın. Probleme göre 2 standart sapma değeri(20) yukarısı aykırı değer olarak kabul edilebilir. Böylece 120 eşik değer(threshold) olarak kabul edilmiş olur.

   Bu aykırı değer hesaplama işlemlerindeki kritik nokta eşik değeri belirlemektir. Kabul edilebilir son nokta neresidir ?

3. Z-Skoru Yaklaşımı

   Burada ilgili değişken standart normal dağılıma uyarlanır. Yani standartlaştırılır.

   Normal dağılıma sahip bir veri setinde, bir gözlemin ortalamadan ne kadar standart sapma saptığını gösterir. Herhangi bir gözlem için z-skoru, gözlemden ortalama çıkartıldıktan sonra standart sapmaya bölünmesiyle hesaplanır.

   $Z = (x - \mu) / \sigma$

   x: Gözlem değeri

   μ: Ortalama değer

   σ: Standart sapma

   Not : Normal dağılım, bir veri setinde medyan değerin ve ortalamanın birbirine çok yakın (idealinde eşit) olması durumudur.

4. Boxplot (Interquartile Range - IQR) Yöntemi

   Tek değişkenli (univarite) yöntemler arasındadır.

Verinin dağılımını çeyrek değerler üzerinden görselleştiren bir grafik türüdür.

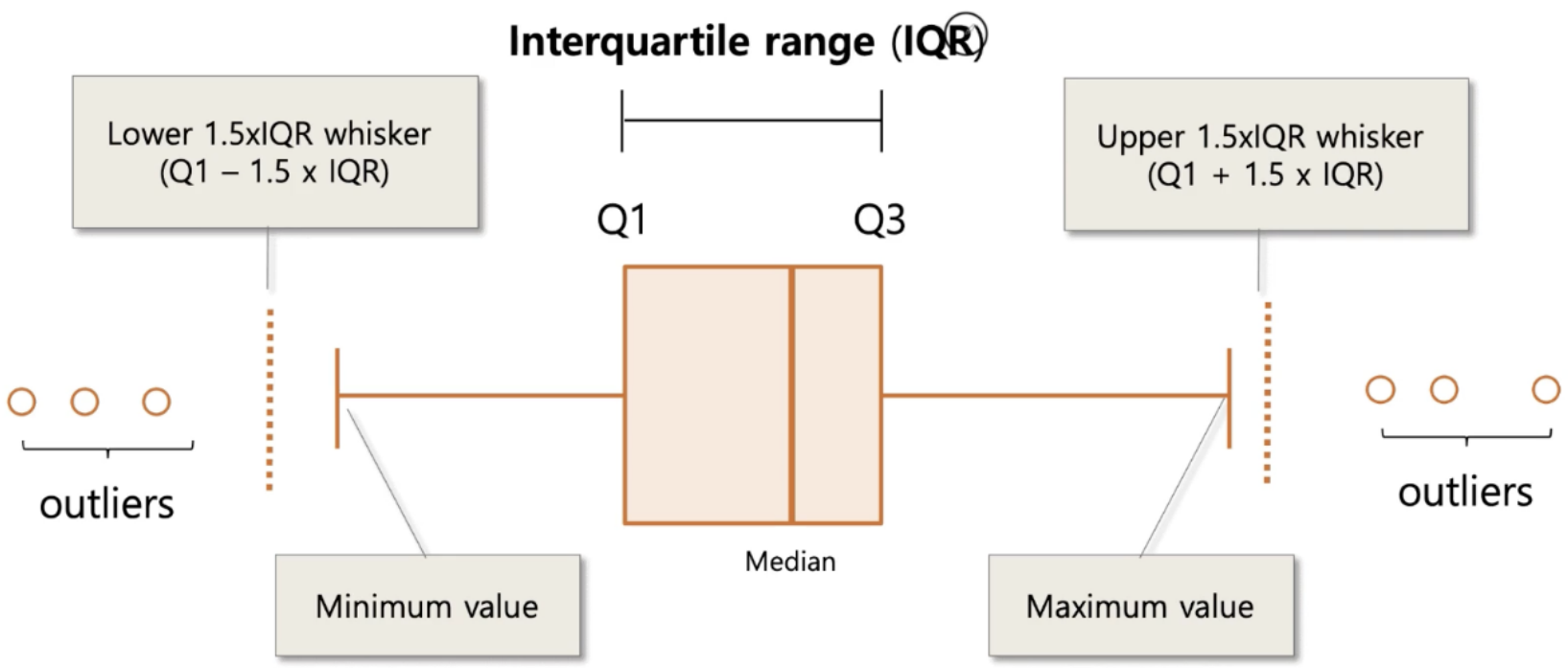
• Kutunun alt ve üst kenarları, 25. ve 75. çeyrek değerlerini temsil eder.

• %75'inci ve %25'inci değerleri arasındaki fark çeyrek açıklığını (IQR), yani orta %50'lik değeri temsil eder.

• Boxplot grafiğinin ortasındaki çizgi, ortanca (median) değeri verir.

• Alt ve üst sınırların dışında kalan noktalar aykırı değerleri temsil eder.

• Alt ve üst sınırların dışında kalan noktalar, yani %25'inci çeyrek değerinden 1.5 kat az ve %75'inci çeyrek değerinden 1.5 kat fazla olan değerler aykırı değerleri temsil ederler.



Örnek boxplot Grafiği ve çeyrekler arası açıklık

Eğer veri setinin içerisinde negatif değerler yoksa, bir değişkende en küçük değer 0 ise, değişkenin değerleri hep pozitifse bu durumda alt sınır çalışmıyor olacaktır. Dolayısıyla çalışmalar üst sınıra yönelik yapılır

Bir boxplot çizildiğinde aykırı değerler görseldeki şekliyle gözlemlenebilir.

Aykırı değer saptamasındaki ana odak, kabul edilebilir eşik değerini belirlemektir. Bu belirlendikten sonra aykırı değerler eşik değerler baz alınarak yakalanırlar.

Buradaki eşik değeri yukarıda belirtilen Standart Sapma Yaklaşımı, Z-Skoru Yaklaşımı, Boxplot (interquartile range - IQR) Yöntemi gibi çeşitli yöntemler ile hesaplanabilir.

Boxplot, bu yöntemlerler arasında en yaygın kullanılan yöntemdir.

5. Çok değişkenli olarak LOF Yöntemi

Aykırı Değerleri Yakalama

install & import işlemleri ve bazı görsel ayarlamalar.

```
# !pip install missingno
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import missingno as msno
from datetime import date
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler, RobustScaler

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.float_format', lambda x: '%3f' % x)
pd.set_option('display.width', 500)
```

Küçük ölçekli örneklerde kullanılmak üzere “titanic” veri seti fonksiyonu, büyük ölçekli uygulamalarda kullanılmak üzere ise “application_train” veri seti fonksiyonu tanımlanmıştır.

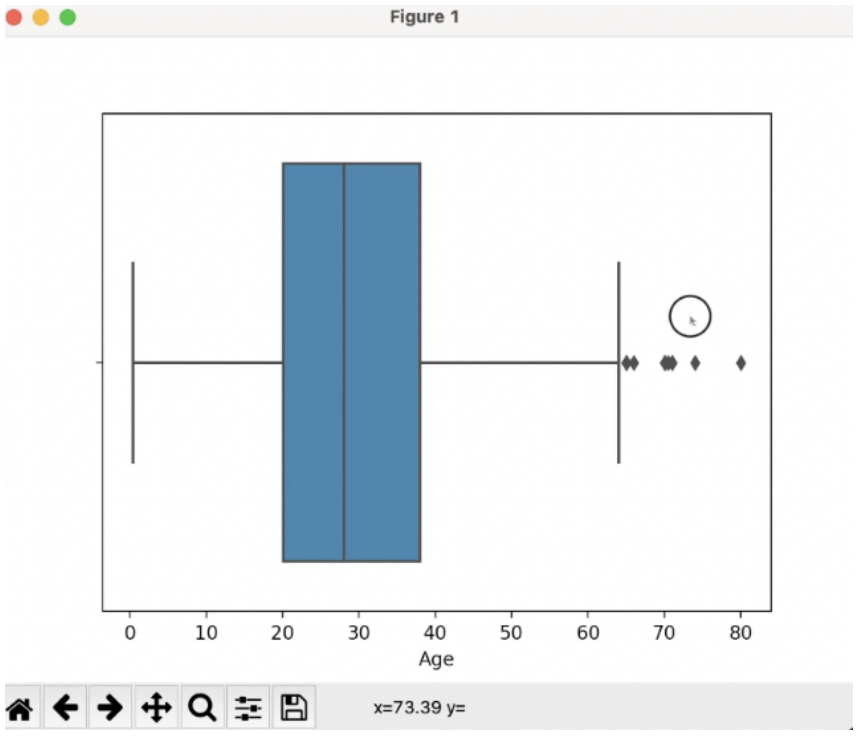
Her seferinde tekrar tekrar veri okuma işlemi yapılmaması için bu fonksiyonlar tanımlanmıştır.

```
def load_application_train():
    data = pd.read_csv("Feature_Engineering/Outliers/dataset/application_train.csv")
    return data


dff = load_application_train()
dff.head()


def load():
    data = pd.read_csv("Feature_Engineering/Outliers/dataset/titanic.csv")
    return data


df = load()
df.head()
```

Grafik tekniği ile aykırı değerler gözlemlenmek istenirse bu durumda boxplot kullanılır. Boxplot sayısal değişkenlerin dağılım bilgisini verir.

```
sns.boxplot(x=df["Age"])
plt.show()
```

Output:



Grafiğin sağ kısmında gözlemlenen, üst sınırların dışında kalan noktalar aykırı değerleri ifade eder.

Aykırı Değerler Nasıl Yakalanır ?

Öncelikle yapılması gereken şey teorik bölümde görülen eşik değerlere erişmektir.

IQR hesabı yapılabilmesi için ilk olarak bir değişkenin çeyreklik değerleri hesaplanmalıdır.

```
# q1 ve q3 çeyrek değerlerinin hesaplanması :
q1 = df["Age"].quantile(0.25)
q3 = df["Age"].quantile(0.75)

# IQR'ın çeyrek değerler ile hesabı :
iqr = q3- q1

# alt ve üst limitin IQR ile hesaplanması :
up = q3 + 1.5 * iqr
low = q1 - 1.5 * iqr

# Alt sınırdan küçük ve üst sınırdan büyük olan değerlere erişmek için :
df[(df["Age"] < low) | (df["Age"] > up)]

# index bilgisine erişmek için :
df[(df["Age"] < low) | (df["Age"] > up)].index

# Aykırı değer olup olmadığını kontrol etmek için :
# any, ifadenin içerisinde herhangi bir değer olup olmadığı sorusuna cevap verir.
# Satır ya da sütuna göre değil, hepsine bakması içim axis=None girilir.
# Buradaki çıktıdan içerisinde bir değer olup olmamasına bağlı olarak True veya False gelir.
df[(df["Age"] < low) | (df["Age"] > up)].any(axis=None)
```

Burada yapılan işlemlerin bir özeti :

1. Eşik değer belirlendi.
2. Aykırı değerlere erişildi.
3. Aykırı değer olup olmadığı sorgulandı.

İşlemleri Fonksiyonlaştırmak

Buradaki en kritik bölüm eşik değer hesaplama bölümüdür.

Dolayısıyla öncelikle outlier_thresholds adında, görevi kendisine girilen değişkenin eşik değerini hesaplamak olan bir fonksiyon tanımlanır.

```
def outlier_thresholds(dataframe, col_name, q1=0.25, q3=0.75):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquantile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquantile_range
    low_limit = quartile1 - 1.5 * interquantile_range
    return low_limit, up_limit


outlier_thresholds(df, "Age")

# "Age" için bu fonksiyonun return'unu low ve up değişkenlerinde tutmak için :
low, up = outlier_thresholds(df, "Age")

# low ve up değerleri elde edildiğine göre seçim işlemi yapılabilir :
df[(df["Age"] < low) | (df["Age"] > up)]
```

Bir değişkenin adı sorulduğunda, o değişken içerisinde aykırı değerler (outliers) olup olmadığının bilgisini veren bir fonksiyon ：

```
def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    if dataframe[(dataframe[col_name] > up_limit) | (dataframe[col_name] < low_limit)].any(axis=None):
        return True
    else:
        return False


check_outlier(df, "Age")
```

Yukarıda oluşturulan outlier_thresholds ve check_outlier fonksiyonlarının programatik bir şekilde veri seti içerisindeki değişkenlere uygulanması gerekmektedir.

Fakat büyük veri setlerinde, çok sayıda değişken olduğu durumlarda bazı problemler ortaya çıkmaktadır.

Bu problemler nümerik ve kategorik değişkenlerin yanı sıra, nümerik görünümlü kategorik ve kategorik görünümlü kardinal değişkenlerin varlığıdır.

Değişken Tipleri	Nedir ?	Örnek Değişkenler	Fonksiyonda Kullanılan İsimleri
Nümerik Değişkenler		Age, Fare	num_cols
Kategorik Değişkenler		Sex, Embarked, Survived, Pclass	cat_cols
Nümerik Görünümlü Kategorik Değişkenler	Kategorik özelliklerini sayısal değerler aracılığı ile ifade eden değişkenlerdir.	Pclass, Survived	num_but_cat
Kategorik Görünümlü Fakat Kardinal	Çok fazla sınıfa sahip olup bilgi taşımayan, seyrekliği çok fazla olan değişkenlerdir.	Name, Ticked, Cabin	cat_but_car

Dolayısıyla, otomatik olarak

• Nümerik

• Kategorik

• Nümerik Görünümlü Kategorik

• Kategorik Görünümlü Fakat Kardinal değişkenlerini ayırt edebilecek bir fonksiyon ihtiyacı vardır.

```
def grab_col_names(dataframe, cat_th=10, car_th=20):
    """

    Veri setindeki kategorik, numerik ve kategorik fakat kardinal değişkenlerin isimlerini verir.
    Not: Kategorik değişkenlerin içerisine numerik görünümlü kategorik değişkenler de dahildir.

    Parameters
    ------
        dataframe: dataframe
                Değişken isimleri alınmak istenilen dataframe
        cat_th: int, optional
                numerik fakat kategorik olan değişkenler için sınıf eşik değeri
        car_th: int, optinal
                kategorik fakat kardinal değişkenler için sınıf eşik değeri

    Returns
    ------
        cat_cols: list
                Kategorik değişken listesi
        num_cols: list
                Numerik değişken listesi
        cat_but_car: list
                Kategorik görünümlü kardinal değişken listesi

    Examples
    ------
        import seaborn as sns
        df = sns.load_dataset("iris")
        print(grab_col_names(df))


    Notes
    ------
        cat_cols + num_cols + cat_but_car = toplam değişken sayısı
        num_but_cat cat_cols'un içerisinde.
        Return olan 3 liste toplamı toplam değişken sayısına eşittir: cat_cols + num_cols + cat_but_car = değişken sayısı

    """
    # Bu fonksiyonda yer alan cat_th ve car_th parametrelerin işlevleri :
    # cat_th=10,  bir sayısal değişken 10'dan az sınıfa sahipse bu değişken kategorik kabul edilir.
    # car_th=20, eğer bir kategorik değişkenin 20'den fazla sınıfı varsa kardinal olarak kabul edilir.

    # cat_cols, cat_but_car
    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == "O"]
    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th and
                   dataframe[col].dtypes == "O"]

    # cat_cols listesi güncellenir :
    cat_cols = cat_cols + num_but_cat  # num_but_cat değişkenleri cat_cols listesine eklenir.
    cat_cols = [col for col in cat_cols if col not in cat_but_car] # kategorik ama kardinal değişkenler cat_cols'tan çıkarılır.

    # num_cols
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != "O"]
    # tipi object'ten farklı fakat kategorik olan değişkenler :
    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th and
                   dataframe[col].dtypes != "O"]
    # tipi object'ten farklı fakat kategorik olan değişkenler num_cols içerisine dahil edilmez :
    num_cols = [ col for col in num_cols if col not in num_but_cat]

    print(f"Observations: {dataframe.shape[0]}")
    print(f"Varaibles: {dataframe.shape[1]}")
    print(f'cat_cols: {len(cat_cols)}')
    print(f'num_cols: {len(num_cols)}')
    print(f'cat_but_car: {len(cat_but_car)}')
    print(f'num_but_cat: {len(num_but_cat)}')
    return cat_cols, num_cols, cat_but_car

# Titanic veri seti(df) için :

# grab_col_names fonksiyonu kullanılır:
cat_cols, num_cols, cat_but_car = grab_col_names(df)

# num_cols içerisindeki "PassengerId" değişkeninin uygun olmadığı düşünüldüğü için çıkarılır:
num_cols = [col for col in num_cols if col not in "PassengerId"]

# check_outlier fonksiyonu kullanılarak outlier gözlemi yapılır:
for col in num_cols:
    print(col, check_outlier(df, col))

# application_train veri seti(dff) için :

# grab_col_names fonksiyonu kullanılır:
cat_cols, num_cols, cat_but_car = grab_col_names(dff)

# num_cols içerisindeki "SK_ID_CURR" değişkeninin uygun olmadığı düşünüldüğü için çıkarılır:
num_cols = [col for col in num_cols if col not in "SK_ID_CURR"]

# check_outlier fonksiyonu kullanılarak outlier gözlemi yapılır:
for col in num_cols:
    print(col, check_outlier(dff, col))
```

Böylece gerçek nümerik değişkenlere, gerçek kategorik değişkenlere ve kardinallere erişilmiş olunur ve `outlier_thresholds` ve `check_outlier` fonksiyonları programatik bir şekilde veri seti içerisindeki değişkenlere uygulanır.

Daha önce manuel bir şekilde aykırı değerler ve index bilgilerine erişilmişti :

• df[(df["Age"] < low) | (df["Age"] > up)]

• df[(df["Age"] < low) | (df["Age"] > up)].index

Aykırı değerler ve index bilgilerine fonksiyonel bir şekilde erişmek için :

```
def grab_outliers(dataframe,col_name, index=False):  # index bilgisinin tercihe göre gelmesi için index parametresi girilir.
    low, up = outlier_thresholds(dataframe, col_name)
    if dataframe[((dataframe[col_name] < low) | (dataframe[col] > up))].shape[0] > 10:  # shape[0] gözlem sayısını, shape[1] değişken sayısını verir. Aykırı değerlerin sayısı kontrol edilir.
        print(dataframe[((dataframe[col_name] < low) | (dataframe[col_name] > up))].head())  # Aykırı değerler 10'un üzerindeyse yalnızca 5 gözlem getirmesi için bir koşul girilir.
    else:  # Aykırı değerler 10'un üzerinde değilse tüm gözlemleri getirmesi için koşul girilir.
        print(dataframe[((dataframe[col_name] < low) | (dataframe[col_name] > up))])

    if index:  # index parametresi True girilirse index bilgilerini return eder.
        outlier_index = dataframe[((dataframe[col_name] < low) | (dataframe[col_name] > up))].index
        return outlier_index


grab_outliers(df, "Age")

grab_outliers(df, "Age", index=True)

# Daha sonra kullanmak üzere saklanmak istenirse :
age_index = grab_outliers(df, "Age", index=True)
```

Özetle bu kısma kadar yapılan önemli işlemler :

```
outlier_thresholds(df, "Age")
check_outlier(df, "Age")
grab_outliers(df, "Age", True)
```

Aykırı Değer Problemini Çözme

Silme İşlemi

Tek bir değişken içerisinde aykırı olmayan değerleri seçmek için :

```
df[~((df["Fare"] < low) | (df["Fare"] > up))]
```

Birden fazla değişken olduğunda, tüm değişkenlerde bulunan aykırı değerleri veri setinden pratik bir şekilde çıkarmak için bu işlemin fonksiyonlaştırılması gerekir.

Silme işleminin fonksiyonlaştırılması :

```
def remove_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    df_without_outliers = dataframe[~((dataframe[col_name] < low_limit) | (dataframe[col_name] > up_limit))]
    return df_without_outliers.
```

“titanic” veri setindeki nümerik değişkenlere aykırı değer silme işemi uygulanmak istenirse :

```
cat_cols, num_cols, cat_but_car = grab_col_names(df)

num_cols = [col for col in num_cols if col not in "PassengerId"]

for col in num_cols:
    new_df = remove_outlier(df, col)

df.shape[0] - new_df.shape[0]  # df ve new_df'in shape[0] farkını alarak ne kadar değişiklik olduğu bilgisine gidilir.
```

Not

Hücredeki bir tane aykırılıktan dolayı silme işlemi gerçekleştirildiğinde, diğer tam olan gözlemlerdeki veriler de kaybedilir. Bu sebeple bazı senaryolarda silme işlemi yerine bu değerleri baskılama yöntemi de tercih edilebilir.

Baskılama Yöntemi (re-assignment with thresholds)

Baskılama yöntemi, kabul edilen eşik değerinin dışında kalan değerlerin(aykırı değerlerin) eşik değerler ile değiştirilmesidir.

Silme işleminden ortaya çıkabilecek veri kaybı dolayısıyla, veri kaybetmeden bu aykırılığı uzaktırmak amaçlı aykırı değerler yakalandıktan sonra eşik değerler ile değiştirilir.

```
# Fare değişkeni için limitler :
low, up = outlier_thresholds(df, "Fare")

# Fare değişkeni için aykırı değerler :
df[((df["Fare"] < low) | (df["Fare"] > up))]["Fare"]  # Sadece Fare column'u gelir.

# Fare değişkeni için aykırı değerlerin loc methodu ile seçilmesi
df.loc[((df["Fare"] < low) | (df["Fare"] > up)), "Fare"]  # Sadece Fare column'u gelir.

# Eşik değerin üzerinde kalan aykırı değerler, "up" eşik değeri ile değiştirilir.
df.loc[(df["Fare"] > up), "Fare"] = up  # up, eşik değeri temsil eder.

# Bu işlem tekrar çalıştırılırsa boş liste gelir:
df.loc[(df["Fare"] > up), "Fare"] # çünkü aykırı değerler eşik değerleri ile değiştirildi.
```

Aynı işlem alt sınıra göre de yapılabilir fakat bu değişkenlerde öyle bir ihtiyaç söz konusu değildir.

Örnek olması amaçlı alt sınır için gereken kod :

```
df.loc[(df["Fare"] < low), "Fare"] = low
```

Bu işlemlerin fonksiyonlaştırılması :

```
def replace_with_thresholds(dataframe, variable):
    low_limit, up_limit = outlier_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit

df = load()  # Veri seti eski haline getirildi.
cat_cols, num_cols, cat_but_car = grab_col_names(df)  # Değişken seçim işlemi gerçekleştirildi.
num_cols = [col for col in num_cols if col not in "PassengerId"]
df.shape

# Outlier kontrolü :
for col in num_cols:
```

```
    print(col, check_outlier(df, col))

# Her bir değişken için threshold'lar ile aykırı değerler değiştirilir.
for col in num_cols:
    replace_with_thresholds(df, col)

# replace_with_thresholds işleminden sonra tekrar outlier kontrolü yapılmak istenirse:
for col in num_cols:
    print(col, check_outlier(df, col))
```

Yapılan İşlemlerin Özeti

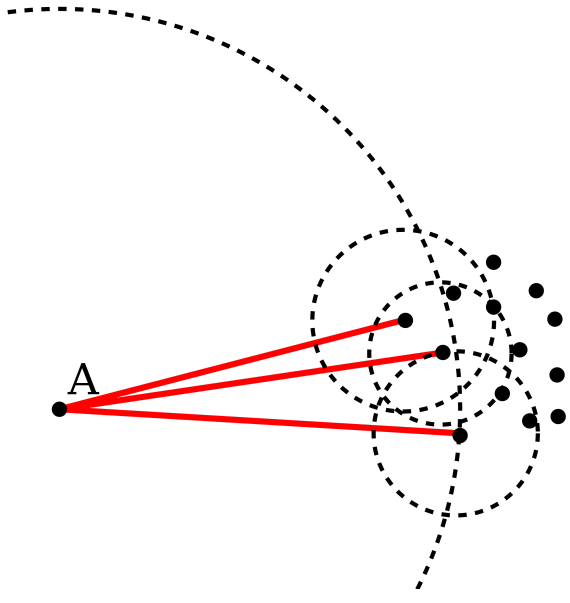
```
df = load()

outlier_thresholds(df, "Age")  # Eşik değeri.
check_outlier(df, "Age")  # Outlier olup olmadığının kontrolü.
grab_outliers(df, "Age", index=True)  # Aykırı değerler ve index bilgilerine erişme işlemi.

remove_outlier(df, "Age")  # Aykırı değerleri veri setinden silme.
replace_with_thresholds(df, "Age")  # Aykırı değerlerin eşik değerler ile değiştirilmesi.(Baskılama Yöntemi)
```

Çok Değişkenli Aykırı Değer Analizi: Local Outlier Factor

Çok Değişkenli Aykırı Değer : Tek başına aykırı kabul edilmeyen bazı değerlerin birlikte ele alındığında bu durumun aykırılık yaratmasıdır.

Bu sebeple aykırı değişkenlerin çok değişkenli olarak da incelenmesi gerekmektedir.

Local Outlier Factor Yöntemi : Gözlemleri, bulundukları konumda yoğunluk tabanlı skorlayarak, buna göre aykırı olabilecek değerleri tanıma imkanı sağlar.



Bir noktanın lokal yoğunluğu, ilgili noktanın etrafındaki komşuluklar anlamına gelir.

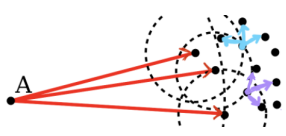
Eğer bir nokta komşularının yoğunluğundan anlamlı bir şekilde düşük ise bu durumda bu nokta daha seyrek bir bölgededir ve bu noktanın aykırı değer olduğu çıkarımı yapılabilir.

Örneğin :

Yukarıdaki şekilden yandaki gibi bir nokta seçilip ele alındığı varsayılsın.

Bu seçilen noktanın etrafındaki yoğunluk incelenir ve bunun için en yakın 3 komşu noktaya bakılır.

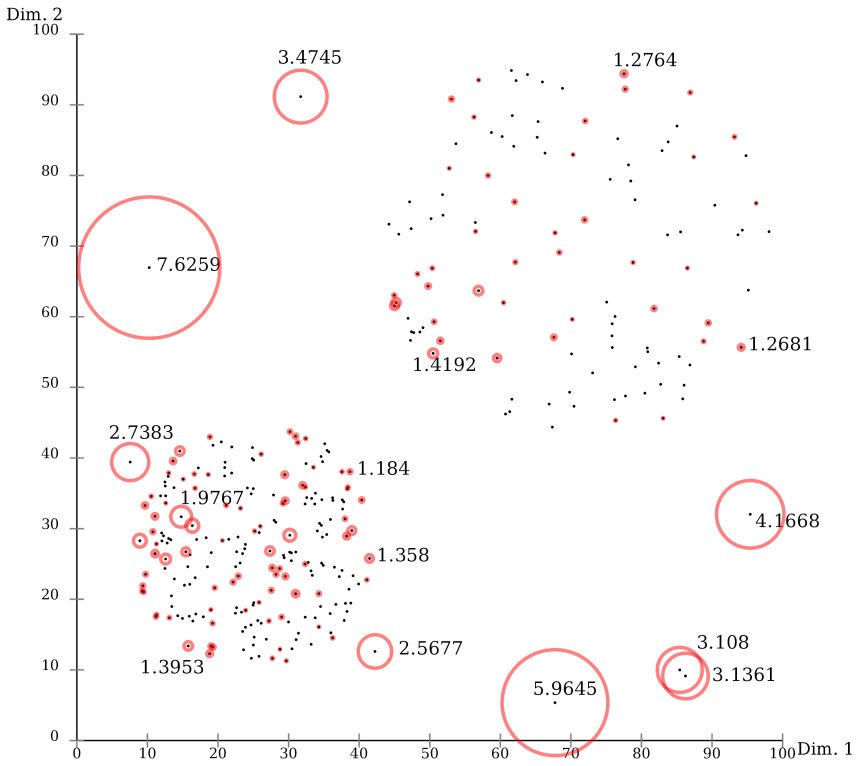
Bu noktaların komşuları arası uzaklık, aşağıdaki her bir nokta için yakın değerlere sahiptir. Yani yoğunluklar birbirine benzerdir.



Fakat diğerlerinden daha uzakta konumlandırılmış A noktası ele alındığında, en yakın 3 komşusu ile mesafesi diğer noktalara göre çok fazladır.

Yani A noktası komşularının lokal yoğunluklarından oldukça farklıdır, anlamlı bir şekilde düşüktür. Bu noktanın aykırı değer olduğu varsayılır.

LOF (Local Outlier Factor) yöntemi, bu komşuluklara göre uzaklık skoru hesaplama imkanı sağlar.



LOF (Local Outlier Factor) Yöntemi 2 Boyutlu Gösterim

LOF her bir nokta için bir skor üretir. 1’in altındaki değerler daha yoğun bölgeyi temsil ederken, 1’den önemli ölçüde daha büyük değerler aykırı değerleri (Outliers) temsil eder.

Örneğin yukarıdaki grafikte skoru 7.6259 olan nokta diğer noktalardan çok uzaktadır ve LOF skoru 1’in çok üzerindedir. Outlier olarak gözlemlenir.

Kırmızı ile işaretlenmemiş noktalar ise LOF skorları 1’e çok yakın olan noktalardır ve outlier olmayan (Inlier) noktalardır.

LOF Yöntemi, bir threshold olarak müdahale etme imkanı sağlar.

2 değişken söz konusu olduğunda bu şekilde 2 boyutta görselleştirilebilir. Çok fazla değişken söz konusu olduğunda(örneğin 100 değişken) 2 boyutta görselleştirme nasıl gerçekleşir ?

Eğer bu 100 değişkeni, 100 değişkenin taşıdığı bilginin büyük bir miktarını taşıdığı varsayılabilecek 2 boyuta indirgenebilirse 2 boyutta görselleştirilebilir. Bu da PCA(Temel Bileşen Analizi) Yöntemi ile gerçekleştirilebilir.

Local Outlier Factor Uygulaması

```
df = sns.load_dataset('diamonds')
df = df.select_dtypes(include=['float64', 'int64'])  # yalnızca sayısal değişkenlerle çalışmak için.
df = df.dropna()
df.head()
df.shape
```

```
# Outlier olup olmadığı kontrolü :
for col in df.columns:
    print(col, check_outlier(df, col))

# "carat" değişkeninin kaç tane outliers'a sahip olduğunu öğrenmek için :
low, up = outlier_thresholds(df, 'carat')
df[((df['carat'] < low) | (df['carat'] > up))].shape

# "depth" değişkeninin kaç tane outliers'a sahip olduğunu öğrenmek için :
low, up = outlier_thresholds(df, 'depth')
df[((df['depth'] < low) | (df['depth'] > up))].shape
```

Çok değişkenli yaklaşıldığında uygulanacak adımlar :

```
clf = LocalOutlierFactor(n_neighbors=20)  # Çalışma başında import edilen LOF methodu çalıştırılır.
clf.fit_predict(df)  # LocalOutlierFactor yöntemi df'e uygulanır. fit_predict LOF skorlarını getirir.
df_scores = clf.negative_outlier_factor_  # df_scores LOF skorlarıdır. Takip edilebilirlik açısından negative_outlier_factor_ negatif skorları getirir.
df_scores[0:5]
# pozitif skorlar için : df_scores = -df_scores
```

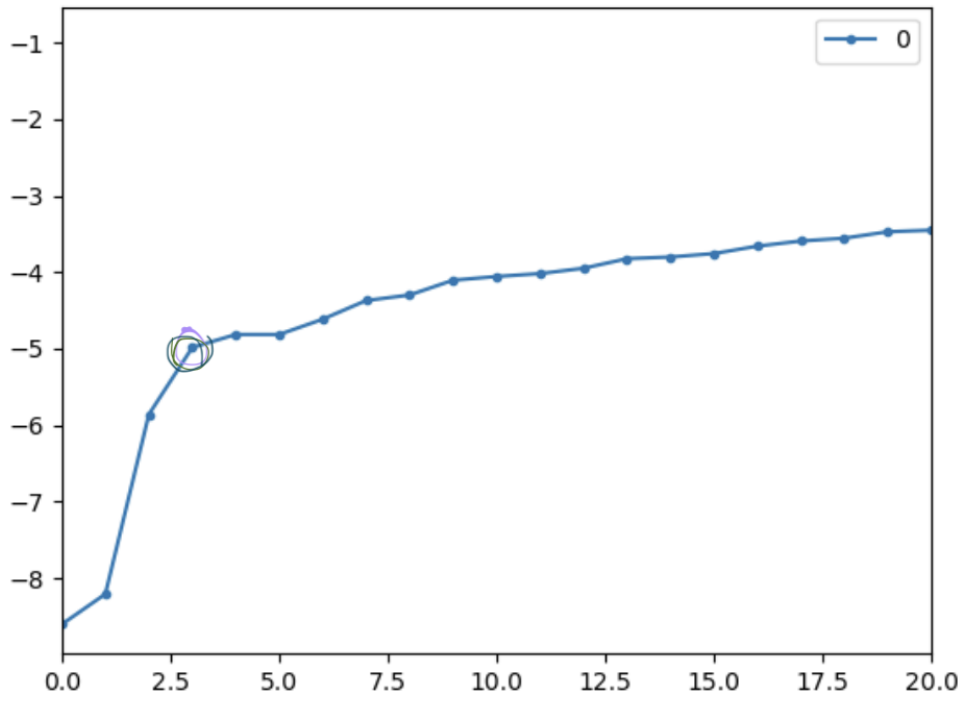
Skorların negative_outlier_factor_ fonksiyonunun sağladığı negatif değerleri ile kullanımasının sebebi :

Eşik değere karar vermek için oluşturulacak olan elbow yöntemi (dirsek yöntemi) grafik tekniğinin daha rahat okunabilmesi açısından negatif olarak kullanılır.

Normal şartlarda LOF skorları 1’e yakınlığı Inlier olma durumunu gösterir fakat burada -1’e yakınlık Inlier olma durumunu gösterir .

LOF skorları -1’den uzaklaştıkça değerler Outlier olma eğiliminde olacaklardır.

```
scores = pd.DataFrame(np.sort(df_scores))  # Skorlar sıralanır.
scores.plot(stacked=True, xlim=[0, 20], style='.-'). # Skorlar görselleştirilir.
plt.show()
```



Y ekseni : Outliers Skorları, X ekseni : Gözlemler

Buradaki her bir nokta olası eşik değerleri temsil etmektedir ve bu olası eşik değerlere göre grafik oluşturulmuştur.

Eşik değer ne olmalı ?

Elbow(Dirsek) Yöntemi :

Elimizde bazı gözlemler ve skorları var. Ve bu skorlara karşılık hangi noktanın eşik değer olması gerektiği problemi var.

Grafikte eğim değişimin fark edildiği en son nokta işaretlenmiştir.

En son değişimin fark edildiği nokta, işaretlenen nokta olduğundan bu nokta eşik değeri olarak belirlenebilir.

İşaretli alanın ilerisinde kalan değerlerde gözle görülür bir değişim yoktur. İşaretli noktanın altında kalanlar ise kötü değerlerdir yorumu yapılabilir.

Bu sebeple 3. indexteki değer eşik değeri kabul edilir.

```
th = np.sort(df_scores)[3]  # threshold 3. indexteki değer olarak belirlendi
# Out[72]: -4.984151747711709
```

Özet :

Outlier hesaplama işlemlerinde ana problem eşik değeri belirlemektir.

Daha önce bu eşik değeri her bir değişkenin kendi içerisinde belirlenmişti.

Burada ise bütün değişkenleri aynı anda değerlendirecek bir yöntem olan LOF yöntemi kullanıldı.

LOF yöntemi, tüm gözlem birimleri için skolar sağladı. Ve bu skorlar arasından bir eşik değeri seçildi.

Bu eşik değer kullanılarak, eşik değerden küçük olanlar, yani aykırı olan değerler seçilebilir.

Burada eşik değeri -4.98’dir. -4.98’den daha küçük olan değerler outlier olarak kabul edilir.

```
df[df_scores < th]  # threshold'tan(-4.98) küçük değerler, yani outlier'ler seçilir.
df[df_scores < th].shape # Out[74]: (3, 7)
```

	carat	depth	table	price	x	y	z
41918	1.030000	78.200000	54.000000	1262	5.720000	5.590000	4.420000
48410	0.510000	61.800000	54.700000	1970	5.120000	5.150000	31.800000
49189	0.510000	61.800000	55.000000	2075	5.150000	31.800000	5.120000

df[df_scores < th].shape ile elde edildiği gibi 3 gözlem gelmiştir.

Bu değerler neden aykırıdır ?

Daha önce tek bir değişken için kontrol edildiğinde çok fazla Outlier değeri ortaya çıktığı gözlemlenmişti.

Burada çok değişkenli etkiye bakıldığında ise yalnıca 3 Outlier değeri gözlemleniyor. Bunun sebebi nedir ?

```
df.describe([0.01, 0.05, 0.75, 0.90, 0.99]).T  # Yukarıdaki sorunun cevabı için değişkenler gözlemlenir.
```

Out[75]:

	count	mean	std	min	1%	5%	50%	75%	90%	99%	max
carat	53940.000000	0.797940	0.474011	0.200000	0.240000	0.300000	0.700000	1.040000	1.510000	2.180000	5.010000
depth	53940.000000	61.749405	1.432621	43.000000	57.900000	59.300000	61.800000	62.500000	63.300000	65.600000	79.000000
table	53940.000000	57.457184	2.234491	43.000000	53.000000	54.000000	57.000000	59.000000	60.000000	64.000000	95.000000
price	53940.000000	3932.799722	3989.439738	326.000000	429.000000	544.000000	2401.000000	5324.250000	9821.000000	17378.220000	18823.000000
x	53940.000000	5.731157	1.121761	0.000000	4.020000	4.290000	5.700000	6.540000	7.310000	8.360000	10.740000
y	53940.000000	5.734526	1.142135	0.000000	4.040000	4.300000	5.710000	6.540000	7.300000	8.340000	58.900000
z	53940.000000	3.538734	0.705699	0.000000	2.480000	2.650000	3.530000	4.040000	4.520000	5.150000	31.800000

Yakalanan aykırı değerlerin index bilgileri edinilir ve sonrasında veri setinden silinebilir.

```
df[df_scores < th].index. # Yakalanan aykırı değerlerin index bilgileri edinilir.

df[df_scores < th].drop(axis=0, labels=df[df_scores < th].index)  # Daha sonra veri setinden silinebilir.
```

Veya Baskılama Yöntemi ile bu Outlier problemi çözülebilir.

Fakat baskılama işlemi hangi eşik değere göre yapılmalı ?

Artık hücrelerde değil gözlem birimleri ile ilgilenildiğinden, gözlem birimine göre baskılama işlemi gerçekleşmesi gerektiğinden, aykırılığı barındıran gözlemi tamamen kaldırıp yerine başka bir gözlem koyulması gerekmektedir.

Bu işlem çoklama kayıt üretecek ve veri bozulmasına sebep olacaktır.

Gözlem sayısı bir miktar fazlaysa baskılama işlemi yapmak problemlere sebep olabilmektedir.

Ağaç yöntemler ile çalışılıyorsa bu değerlere hiç dokunulmamalıdır.

Yapılacak max işlem veri setini bozmadan en uç değerleri temizlemek olmalıdır. (IQR %5-%95 veya %1-%99) Bunun için outlier_threshold ve replace_with_threshold kullanılarak gözlemlere kendi içinde yaklaşılır.

References:

• https://courses.miuul.com/p/feature-engineering?gclid=Cj0KCQiA1ZGcBhCoARIsAGQ0kkrOv2WlbA52S5pmw4iGYG5vLYUsMOEdaCWe79tQQlKqfHq5vk1IhV0aAjT2EALw_wcB

• https://ravenfo.com/2021/02/11/aykiri-deger-analizi/

• https://www.veribilimiokulu.com/local-outlier-factor-ile-anormallik-tespiti/

• https://www.veribilimiokulu.com/makine-ogrenmesine-cok-degiskenli-istatistiksel-yaklasimlar-temel-bilesenler-analizi/