

PyOD, verilerinizdeki aykırı değerleri tespit etmek için böyle bir kitaplıktır. Aykırı değerleri tespit etmek için 20'den fazla farklı algoritmaya erişim sağlar ve hem Python 2 hem de 3 ile uyumludur. Mutlak bir mücevher!

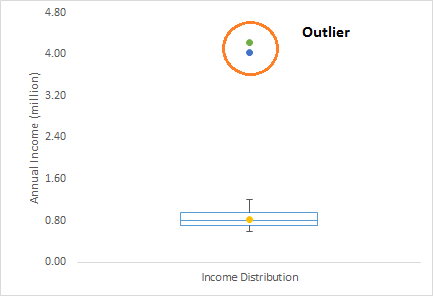
Aykırı değerlerin var olmasının birçok nedeni vardır. Belki de bir analist veri girişinde bir hata yaptı ya da makine ölçümde bir hata yaptı ya da aykırı değer kasıtlı bile olabilir! Bazı insanlar bilgilerini ifşa etmek istemezler ve bu nedenle formlara yanlış bilgiler girerler.

Aykırı değerler iki türdendir: **Tek Değişkenli**ve **Çok Değişkenli**. Tek değişkenli bir aykırı değer, yalnızca bir değişkendeki uç değerlerden oluşan bir veri noktasıdır, oysa çok değişkenli bir aykırı değer, en az iki değişken üzerinde birleşik olağandışı bir puandır

### Aykırı Değer nedir?

Aykırı değer, analistler ve veri bilimcileri tarafından yaygın olarak kullanılan bir terminolojidir, çünkü yakından ilgilenilmesi gerekir, aksi takdirde çılgınca yanlış tahminlere neden olabilir. Basitçe söylemek gerekirse, Outlier, çok uzakta görünen ve bir örneklemdeki genel modelden farklılaşan bir gözlemdir.

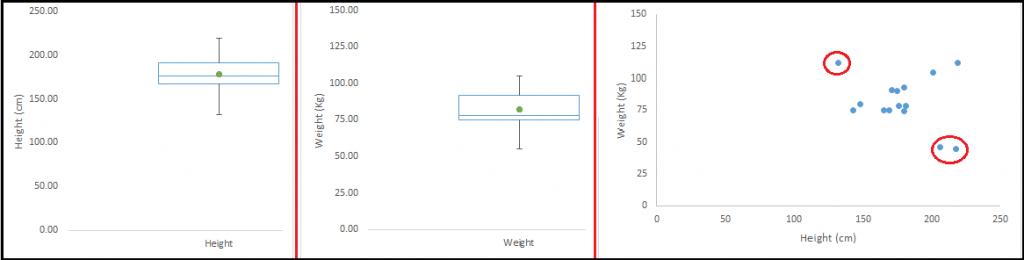
Bir örnek verelim, müşteri profili çıkaralım ve müşterilerin ortalama yıllık gelirinin 0,8 milyon dolar olduğunu öğrenelim. Ancak, yıllık geliri 4 dolar ve 4.2 milyon dolar olan iki müşteri var. Bu iki müşterinin yıllık geliri, nüfusun geri kalanından çok daha yüksektir. Bu iki gözlem Aykırı Değerler olarak görülecektir.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Outlier.png)

### Aykırı Değer türleri nelerdir?

Aykırı değer iki türde olabilir:  **Tek Değişkenli** ve  **Çok Değişkenli** . Yukarıda, tek değişkenli aykırı değer örneğini tartıştık. Bu aykırı değerler, tek bir değişkenin dağılımına baktığımızda bulunabilir. Çok değişkenli aykırı değerler, n boyutlu bir uzayda aykırı değerlerdir. Bunları bulmak için çok boyutlu dağılımlara bakmanız gerekir.

Bunu bir örnekle anlayalım. Boy ve kilo arasındaki ilişkiyi anladığımızı varsayalım. Aşağıda Boy, Kilo için tek değişkenli ve iki değişkenli dağılımımız var. Kutu grafiğine bir göz atın. Aykırı değere sahip değiliz (1.5 \* IQR'nin üstünde ve altında, en yaygın yöntem). Şimdi dağılım grafiğine bakın. Burada, belirli bir ağırlık ve boy segmentinde ortalamanın altında ve bir üstünde iki değerimiz var.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Outlier_21.png)

**Aykırı değerlere ne sebep olur?**

Aykırı değerlerle karşılaştığımızda, onlarla baş etmenin ideal yolu, bu aykırı değerlere sahip olmanın nedenini bulmaktır. Onlarla başa çıkma yöntemi, daha sonra oluşum nedenlerine bağlı olacaktır. Aykırı değerlerin nedenleri iki geniş kategoride sınıflandırılabilir:

1. **Yapay (Hata) / Doğal Değil**
2. **Doğal** .

Çeşitli aykırı değerleri daha ayrıntılı olarak anlayalım:

* **Veri Giriş Hataları: -**  Veri toplama, kaydetme veya giriş sırasında ortaya çıkan hatalar gibi insan hataları, verilerde aykırı değerlere neden olabilir. Örneğin: Bir müşterinin yıllık geliri 100.000 $ 'dır. Yanlışlıkla, veri giriş operatörü şekle ek bir sıfır koyar. Şimdi gelir 10 kat daha yüksek olan 1.000.000 $ olur. Açıkça, bu, nüfusun geri kalanıyla karşılaştırıldığında aykırı değer olacaktır.
* **Ölçüm Hatası:** Aykırı değerlerin en yaygın kaynağıdır. Bu, kullanılan ölçüm aletinin arızalı olduğu ortaya çıktığında ortaya çıkar. Örneğin: 10 tartı makinesi var. 9 tanesi doğru, 1 tanesi hatalı. Arızalı makinedeki kişiler tarafından ölçülen ağırlık, gruptaki diğer insanlardan daha yüksek / daha düşük olacaktır. Hatalı makinede ölçülen ağırlıklar aykırı değerlere yol açabilir.
* **Deneysel Hata:** Aykırı değerlerin bir başka nedeni de deneysel hatadır. Örneğin: 7 koşucudan oluşan 100 metrelik bir sprintte, bir koşucu geç başlamasına neden olan 'Başla' çağrısına konsantre olmayı kaçırdı. Dolayısıyla bu, koşucunun koşma süresinin diğer koşuculardan daha fazla olmasına neden oldu. Toplam çalışma süresi bir aykırı değer olabilir.
* **Kasıtlı Aykırı Değer *:*** Bu genellikle hassas verileri içeren, kendi kendine bildirilen ölçümlerde bulunur. Örneğin: Gençler genellikle tükettikleri alkol miktarını rapor ederler. Sadece bir kısmı gerçek değeri bildirebilir. Burada gerçek değerler aykırı değerler gibi görünebilir çünkü gençlerin geri kalanı tüketimi rapor etmiyor.
* **Veri İşleme Hatası:** Veri madenciliği yaptığımızda, birden çok kaynaktan veri çıkarıyoruz. Bazı manipülasyon veya çıkarma hatalarının veri setinde aykırı değerlere yol açması mümkündür.
* **Örnekleme hatası:**  Örneğin, sporcuların boyunu ölçmemiz gerekiyor. Yanlışlıkla, örneğe birkaç basketbolcuyu dahil ettik. Bu dahil etme, veri setinde aykırı değerlere neden olabilir.
* **Doğal Aykırı Değer:** Aykırı değer yapay olmadığında (hata nedeniyle), bu doğal bir aykırıdır. Örneğin: Ünlü sigorta şirketlerinden biriyle yaptığım son görevde, en iyi 50 mali danışmanın performansının nüfusun geri kalanından çok daha yüksek olduğunu fark ettim. Şaşırtıcı bir şekilde, herhangi bir hatadan kaynaklanmadı. Bu nedenle, danışmanlarla herhangi bir veri madenciliği faaliyeti gerçekleştirdiğimizde, bu bölümü ayrı ayrı ele alırdık.

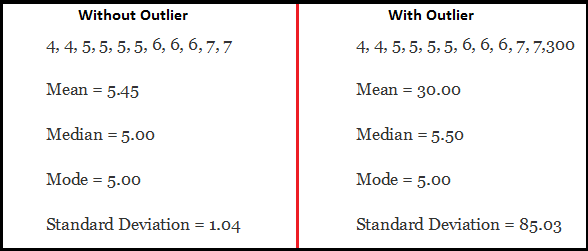
**Aykırı Değerlerin bir veri kümesi üzerindeki etkisi nedir?**

Aykırı değerler, veri analizi ve istatistiksel modellemenin sonuçlarını büyük ölçüde değiştirebilir. Veri setinde aykırı değerlerin çok sayıda olumsuz etkisi vardır:

* Hata varyansını artırır ve istatistiksel testlerin gücünü azaltır
* Aykırı değerler rastgele dağıtılmazsa, normalliği azaltabilirler
* Önemli ölçüde ilgi çekici olabilecek tahminleri önyargılı olabilir veya etkileyebilirler.
* Ayrıca, Regresyon, ANOVA ve diğer istatistiksel model varsayımlarının temel varsayımını da etkileyebilirler.

Etkiyi derinlemesine anlamak için, veri kümesinde aykırı değerler olan ve olmayan bir veri kümesine ne olduğunu kontrol etmek için bir örnek verelim.

**Misal:**

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Outlier_31.png)

Gördüğünüz gibi, aykırı değerlere sahip veri seti, önemli ölçüde farklı ortalama ve standart sapmaya sahiptir. İlk senaryoda ortalamanın 5,45 olduğunu söyleyeceğiz. Ancak aykırı değerle ortalama 30'a yükselir. Bu, tahmini tamamen değiştirir.

**Aykırı Değerler nasıl tespit edilir?**

Aykırı değerleri tespit etmek için en yaygın kullanılan yöntem görselleştirmedir. **Box-plot** , **Histogram** , **Scatter Plot** gibi çeşitli görselleştirme yöntemleri kullanıyoruz (yukarıda, görselleştirme için kutu grafiği ve dağılım grafiği kullandık). Bazı analistler, aykırı değerleri tespit etmek için çeşitli pratik kurallar da kullanır. Onlardan bazıları:

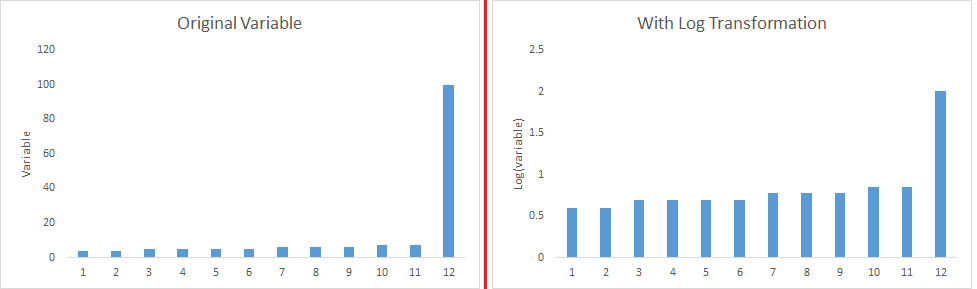
* -1,5 x IQR ile 1,5 x IQR aralığının ötesinde herhangi bir değer
* Sınırlama yöntemlerini kullanın. 5. ve 95. yüzdelik aralığın dışında kalan herhangi bir değer, aykırı değer olarak kabul edilebilir
* Veri noktaları, ortalamadan üç veya daha fazla standart sapma, aykırı değer olarak kabul edilir
* Aykırı değer tespiti, yalnızca etkili veri noktaları için verilerin incelenmesinin özel bir durumudur ve aynı zamanda iş anlayışına da bağlıdır.
* İki değişkenli ve çok değişkenli aykırı değerler tipik olarak bir etki indeksi veya kaldıraç veya mesafe kullanılarak ölçülür. Mahalanobis'in mesafesi ve Cook's *D* gibi popüler endeksler , aykırı değerleri tespit etmek için sıklıkla kullanılır.
* SAS'da PROC Univariate, PROC SGPLOT kullanabiliriz. Aykırı değerleri ve etkili gözlemi belirlemek için, STUDENT, COOKD, RSTUDENT ve diğerleri gibi istatistiksel ölçülere de bakarız.

**Aykırı Değerler nasıl kaldırılır?**

Aykırı değerlerle başa çıkmanın yollarının çoğu, gözlemleri silme, dönüştürme, ikiye ayırma, ayrı bir grup olarak işleme, değerleri atama ve diğer istatistiksel yöntemler gibi eksik değer yöntemlerine benzer. Burada, aykırı değerlerle başa çıkmak için kullanılan yaygın teknikleri tartışacağız:

**Gözlemlerin silinmesi:** Veri giriş hatası, veri işleme hatası veya aykırı değer gözlemleri sayı olarak çok küçükse aykırı değerleri sileriz. Aykırı değerleri gidermek için her iki uçta da kırpmayı kullanabiliriz.

**Değerleri dönüştürme ve gruplama:** Değişkenleri dönüştürmek, aykırı değerleri de ortadan kaldırabilir. Bir değerin doğal logaritması, aşırı değerlerin neden olduğu değişimi azaltır. Binning ayrıca bir değişken dönüşüm biçimidir. Karar Ağacı algoritması, değişkenlerin gruplanması nedeniyle aykırı değerlerle iyi başa çıkmayı sağlar. Farklı gözlemlere ağırlık atama sürecini de kullanabiliriz.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Transformation_1.png)

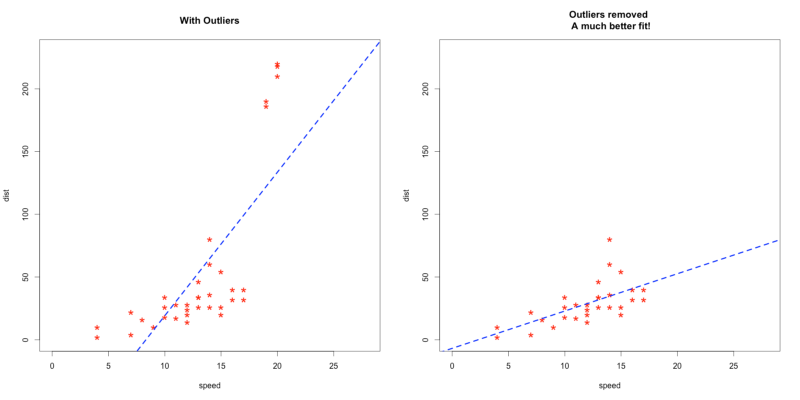
**İmputing:** gibi  [eksik değerler isnat](https://www.analyticsvidhya.com/blog/2015/02/7-steps-data-exploration-preparation-building-model-part-2/) , biz de atfetmek aykırı. Ortalama, medyan, mod isnat yöntemlerini kullanabiliriz. Değerleri yüklemeden önce, bunun doğal aykırı değer mi yoksa yapay mı olduğunu analiz etmeliyiz. Yapay ise, empoze edici değerler ile gidebiliriz. Aykırı değer gözlemlerinin değerlerini tahmin etmek için istatistiksel modeli de kullanabiliriz ve bundan sonra onu tahmin edilen değerlerle ilişkilendirebiliriz.

**Ayrı ayrı ele alın:** Önemli sayıda aykırı değer varsa, bunları istatistiksel modelde ayrı ayrı ele almalıyız. Yaklaşımlardan biri, her iki grubu iki farklı grup olarak ele almak ve her iki grup için ayrı bir model oluşturmak ve ardından çıktıyı birleştirmektir.

Buraya kadar, veri keşfi adımları, eksik değer işleme ve aykırı değer tespiti ve işleme teknikleri hakkında bilgi edindik. Bu 3 aşama, ham verilerinizi bilgi kullanılabilirliği ve doğruluğu açısından daha iyi hale getirecektir. Şimdi veri keşfinin son aşamasına geçelim. Özellik Mühendisliğidir.

## Aykırı Değerleri Neden Tespit Etmeliyiz?

Aykırı değerler, analizimizin ve istatistiksel modellememizin sonuçlarını ciddi bir şekilde etkileyebilir. Aykırı değerler mevcut olduğunda ve ele alındıklarında bir modele ne olduğunu görselleştirmek için aşağıdaki resme bakın:



**Ancak uyarı şudur - aykırı değerler her zaman kötü bir şey değildir.**Bunu anlamak çok önemli. Aykırı değerleri, sonuçları nasıl etkileyeceklerini düşünmeden verilerinizden kaldırmak bir felaket reçetesidir.

*"Aykırı değerler mutlaka kötü bir şey değildir. Bunlar sadece diğerleriyle aynı modeli takip etmeyen gözlemlerdir. Ancak bir aykırı değer çok ilginç olabilir. Örneğin, biyolojik bir deneyde, bir sıçan ölmemişken diğerleri ölüyse, nedenini anlamak çok ilginç olacaktır. Bu, yeni bilimsel keşiflere yol açabilir. Bu nedenle, aykırı değerleri tespit etmek önemlidir. "*

*- Pierre Lafaye de Micheaux, Yazar ve İstatistikçi*

Eğilimimiz, aykırı değerleri tespit etmek için kutu grafikleri, histogramlar ve dağılım grafikleri gibi basit yöntemler kullanmaktır. Ancak **özel aykırı değer algılama algoritmaları, büyük miktarda veriyi işleyen ve daha büyük veri kümelerinde model tanıma gerçekleştirmek için bir araç gerektiren alanlarda son derece değerlidir.**

Finans alanında dolandırıcılık tespiti ve ağ güvenliğinde izinsiz giriş tespiti gibi uygulamalar, aykırı değerleri tespit etmek için yoğun ve doğru teknikler gerektirir. Aykırı bir değer tespit etseniz ve bunun gerçek olduğu ortaya çıkarsa ne kadar utanç verici olacağını hayal edebiliyor musunuz?

PyOD kitaplığı bu boşluğu kapatmak için devreye girebilir. Bakalım her şey neyle ilgili.

## Aykırı Değer Tespiti için Neden PyOD Kullanmalıyız?

Çeşitli programlama dillerinde çok sayıda aykırı değer algılama paketi mevcuttur. Bu dilleri özellikle R'de yararlı buldum. Ancak Python'a geçtiğimde, aykırı değer algılama kitaplığının göze çarpan bir eksikliği vardı. Bu nasıl mümkün oldu ?!

[PyNomaly](https://pypi.org/project/PyNomaly/) gibi mevcut uygulamalar  , aykırı değer tespiti için özel olarak tasarlanmamıştır (yine de kontrol etmeye değer!). Bu boşluğu doldurmak için  Yue Zhao , Zain Nasrullah ve Zheng Li  , PyOD kütüphanesini tasarladı ve uyguladı.

[**PyOD**](https://github.com/yzhao062/pyod)**, çok değişkenli verilerdeki aykırı değerleri tespit etmek için ölçeklenebilir bir Python araç setidir. İyi**[**belgelenmiş**](https://pyod.readthedocs.io/en/latest/)**tek bir API altında yaklaşık 20 aykırı değer algılama algoritmasına erişim sağlar .**

## PyOD'un Özellikleri

PyOD'un birçok avantajı vardır ve pek çok kullanışlı özellikle birlikte gelir. İşte gruptan aldığım seçim:

* Çeşitli algoritmalarda **ayrıntılı belgeler ve örnekler içeren Açık Kaynak**
* **S gelişmiş modeller upports** Sinir Ağları, Derin Öğrenme ve Aykırı Toplulukları dahil
* **JIT** (Tam Zamanında) **ile optimize edilmiş performans** ve numba ve joblib kullanarak paralelleştirme
* **Hem Python 2 hem de 3 ile uyumlu**

## PyOD'u Python'a Yükleme

Python dizüstü bilgisayarlarımızı güçlendirme zamanı! Önce PyOD'u makinelerimize kuralım:

pip install pyod

pip install --upgrade pyod   # en son sürümün kurulu olduğundan emin olmak için!

Kadar basit!

PyOD'un ayrıca Keras'ta uygulanan bazı sinir ağı tabanlı modelleri içerdiğini unutmayın. PyOD olacak  ***DEĞİL*** yüklemek ***keras*** veya ***TensorFlow*** otomatik. Sinir ağı tabanlı modeller kullanmak istiyorsanız, Keras ve diğer kitaplıkları manuel olarak kurmanız gerekecektir.

## PyOD'da kullanılan aykırı değer Algılama Algoritmaları

PyOD'a güç veren aykırı değer algılama algoritmalarını görelim. PyOD'u uygulamak iyi ve güzel ama altında nasıl çalıştığını anlamanın da aynı derecede önemli olduğunu düşünüyorum. Bu, onu bir veri kümesinde kullanırken size daha fazla esneklik sağlayacaktır.

Not:  Bu bölümde bir terim ***Outlying puanı*** kullanacağız . Bu, her modelin bir şekilde, noktanın aykırı olup olmadığını belirlemek için eşik değerini kullanmak yerine bir veri noktasını puanladığı anlamına gelir.

### Angle-Based Outlier Detection (ABOD) : Açı Bazlı Aykırı Değer Tespiti (ABOD)

* Her nokta ile komşuları arasındaki ilişkiyi dikkate alır. *Bu komşular arasındaki ilişkileri dikkate almıyor.*Ağırlıklı kosinüs puanlarının tüm komşulara olan varyansı, dıştaki puan olarak görülebilir.
* ABOD çok boyutlu veriler üzerinde iyi performans gösteriyor
* PyOD, ABOD'nin iki farklı sürümünü sağlar:
  + **Hızlı ABOD:** Yaklaşık olarak k-en yakın komşuları kullanır
  + **Orijinal ABOD:** Yüksek zaman karmaşıklığına sahip tüm eğitim noktalarını dikkate alır

### k-En Yakın Komşular Dedektörü : k-Nearest Neighbors Detector

* Herhangi bir veri noktası için, k'inci en yakın komşusuna olan uzaklık, dıştaki puan olarak görülebilir.
* PyOD, üç [kNN](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/) dedektörünü destekler :
  + **En büyük:** Aykırı değer olarak k'inci komşunun mesafesini kullanır
  + **Ortalama:** Uç değer olarak tüm k komşularının ortalamasını kullanır
  + **Medyan:** Aykırı puan olarak k komşuya olan mesafenin medyanını kullanır

### Isolation Forest : İzolasyon Ormanları

* Scikit-learn kütüphanesini dahili olarak kullanır. Bu yöntemde, veri bölümleme bir dizi ağaç kullanılarak yapılır. İzolasyon Ormanı, noktanın yapı içinde ne kadar izole olduğuna bakan bir anormallik puanı sağlar. Anormallik puanı daha sonra normal gözlemlerden aykırı değerleri belirlemek için kullanılır
* İzolasyon Ormanı çok boyutlu veriler üzerinde iyi performans gösteriyor

### Histogram-based Outlier Detection : Histogram tabanlı Aykırı Değer Tespiti

* Özellik bağımsızlığını varsayan ve histogramlar oluşturarak aykırı değer puanını hesaplayan etkili, denetimsiz bir yöntemdir.
* Çok değişkenli yaklaşımlardan çok daha hızlıdır, ancak daha az kesinlik pahasına

### Yerel Korelasyon İntegrali (LOCI) : Local Correlation Integral (LOCI)

* LOCI, aykırı değerleri ve uç değer gruplarını tespit etmek için çok etkilidir. Her nokta için, noktanın etrafındaki alandaki verilerle ilgili birçok bilgiyi özetleyen, kümeleri, mikro kümeleri, çaplarını ve küme arası mesafelerini belirleyen bir LOCI grafiği sağlar.
* Mevcut aykırı değer algılama yöntemlerinden hiçbiri bu özellikle eşleşemez çünkü her nokta için yalnızca tek bir sayı verirler

### 

### Feature Bagging : Özellik Torbalama

* Bir özellik torbalama dedektörü, veri kümesinin çeşitli alt örneklerine bir dizi temel dedektöre uyar. Tahmin doğruluğunu iyileştirmek için ortalama alma veya diğer kombinasyon yöntemlerini kullanır
* Varsayılan olarak, Yerel Aykırı Değer Faktörü (LOF) temel tahmin aracı olarak kullanılır. Bununla birlikte, kNN ve ABOD gibi herhangi bir tahmin edici, temel tahmin edici olarak kullanılabilir.
* Özellik torbalama ilk olarak , özelliklerin bir alt kümesini rastgele seçerek *n* alt örnek oluşturur. Bu, temel tahmin edicilerin çeşitliliğini ortaya çıkarır. Son olarak, tahmin puanı, tüm temel dedektörlerin ortalamasının alınması veya maksimumunun alınmasıyla oluşturulur.

### Kümeleme Tabanlı Yerel Aykırı Değer Faktörü : Clustering Based Local Outlier Factor

* Verileri küçük kümeler ve büyük kümeler halinde sınıflandırır. Anormallik puanı daha sonra noktanın ait olduğu kümenin büyüklüğüne ve en yakın büyük kümeye olan mesafeye göre hesaplanır.

## PyOD tarafından sağlanan Ekstra Yardımcı Programlar

* Bir fonksiyon **generate\_data**aykırı olan rasgele verileri oluşturmak için de kullanılabilir. Inliers verileri çok değişkenli bir Gauss dağılımı ile oluşturulur ve aykırı değerler [tek tip bir dağılımla oluşturulur](https://www.analyticsvidhya.com/blog/2017/09/6-probability-distributions-data-science/) .
* Aykırı değerler fraksiyonu için kendi değerlerimizi ve veri setimizde olmasını istediğimiz toplam numune sayısını sağlayabiliriz. Bu yardımcı program işlevini uygulama bölümünde veri oluşturmak için kullanacağız

### PyOD on a Simulated Dataset

First, let’s import the required libraries:

import numpy as np

from scipy import stats

import matplotlib.pyplot as plt

%matplotlib inline

import matplotlib.font\_manager

Now, we’ll import the models we want to use to detect the outliers in our dataset. We will be using ABOD (Angle Based Outlier Detector) and KNN (K Nearest Neighbors):

from pyod.models.abod import ABOD

from pyod.models.knn import KNN

Şimdi, aykırı değerlerle rastgele bir veri kümesi oluşturup grafiğini çizeceğiz.

from pyod.utils.data import generate\_data, get\_outliers\_inliers

#generate random data with two features

X\_train, Y\_train = generate\_data(n\_train=200,train\_only=True, n\_features=2)

# by default the outlier fraction is 0.1 in generate data function

outlier\_fraction = 0.1

# store outliers and inliers in different numpy arrays

x\_outliers, x\_inliers = get\_outliers\_inliers(X\_train,Y\_train)

n\_inliers = len(x\_inliers)

n\_outliers = len(x\_outliers)

#separate the two features and use it to plot the data

F1 = X\_train[:,[0]].reshape(-1,1)

F2 = X\_train[:,[1]].reshape(-1,1)

# create a meshgrid

xx , yy = np.meshgrid(np.linspace(-10, 10, 200), np.linspace(-10, 10, 200))

# scatter plot

plt.scatter(F1,F2)

plt.xlabel('F1')

plt.ylabel('F2')

Bir sözlük oluşturun ve aykırı değerleri tespit etmek için kullanmak istediğiniz tüm modelleri ekleyin:

classifiers = {

    'Angle-based Outlier Detector (ABOD)'   : ABOD(contamination=outlier\_fraction),

    'K Nearest Neighbors (KNN)' :  KNN(contamination=outlier\_fraction)

}

Sözlüğe eklediğimiz her modele verileri sığdırın, Ardından her modelin aykırı değerleri nasıl tespit ettiğini görün:

#set the figure size

plt.figure(figsize=(10, 10))

for i, (clf\_name,clf) in enumerate(classifiers.items()) :

# fit the dataset to the model

clf.fit(X\_train)

# predict raw anomaly score

scores\_pred = clf.decision\_function(X\_train)\*-1

# prediction of a datapoint category outlier or inlier

y\_pred = clf.predict(X\_train)

# no of errors in prediction

n\_errors = (y\_pred != Y\_train).sum()

print('No of Errors : ',clf\_name, n\_errors)

# rest of the code is to create the visualization

# threshold value to consider a datapoint inlier or outlier

threshold = stats.scoreatpercentile(scores\_pred,100 \*outlier\_fraction)

# decision function calculates the raw anomaly score for every point

Z = clf.decision\_function(np.c\_[xx.ravel(), yy.ravel()]) \* -1

Z = Z.reshape(xx.shape)

subplot = plt.subplot(1, 2, i + 1)

# fill blue colormap from minimum anomaly score to threshold value

subplot.contourf(xx, yy, Z, levels = np.linspace(Z.min(), threshold, 10),cmap=plt.cm.Blues\_r)

# draw red contour line where anomaly score is equal to threshold

a = subplot.contour(xx, yy, Z, levels=[threshold],linewidths=2, colors='red')

# fill orange contour lines where range of anomaly score is from threshold to maximum anomaly score

subplot.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')

# scatter plot of inliers with white dots

b = subplot.scatter(X\_train[:-n\_outliers, 0], X\_train[:-n\_outliers, 1], c='white',s=20, edgecolor='k')

# scatter plot of outliers with black dots

c = subplot.scatter(X\_train[-n\_outliers:, 0], X\_train[-n\_outliers:, 1], c='black',s=20, edgecolor='k')

subplot.axis('tight')

subplot.legend(

[a.collections[0], b, c],

['learned decision function', 'true inliers', 'true outliers'],

       prop=matplotlib.font\_manager.FontProperties(size=10),

       loc='lower right')

subplot.set\_title(clf\_name)

subplot.set\_xlim((-10, 10))

subplot.set\_ylim((-10, 10))

plt.show()

### PyOD on the Big Mart Sales Problem

import pandas as pd

import numpy as np

# Import models

from pyod.models.abod import ABOD

from pyod.models.cblof import CBLOF

from pyod.models.feature\_bagging import FeatureBagging

from pyod.models.hbos import HBOS

from pyod.models.iforest import IForest

from pyod.models.knn import KNN

from pyod.models.lof import LOF

# reading the big mart sales training data

df = pd.read\_csv("train.csv")

Scatter plot ile çizdirme

df.plot.scatter('Item\_MRP','Item\_Outlet\_Sales')

Öğe Outlet Satışları aralığı 0 ila 12000 arasındadır ve Öğe MRP'si 0 ila 250 arasındadır. Her iki özelliği de 0 ila 1 arasında bir aralığa indireceğiz. Bu, açıklanabilir bir görselleştirme oluşturmak için gereklidir (çok uzatılmış olacaktır. aksi takdirde). Bu verilere gelince, aynı yaklaşımı kullanmak görselleştirmeyi oluşturmak için çok daha fazla zaman alacaktır.

Not: Görselleştirmeyi istemiyorsanız, bir noktanın aykırı olup olmadığını tahmin etmek için aynı ölçeği kullanabilirsiniz.

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(0, 1))

df[['Item\_MRP','Item\_Outlet\_Sales']] = scaler.fit\_transform(df[['Item\_MRP','Item\_Outlet\_Sales']])

df[['Item\_MRP','Item\_Outlet\_Sales']].head()

Bu değerleri daha sonra modellerimizde kullanmak için N *umPy* dizisinde *saklayın* :

X1 = df['Item\_MRP'].values.reshape(-1,1)

X2 = df['Item\_Outlet\_Sales'].values.reshape(-1,1)

X = np.concatenate((X1,X2),axis=1)

Yine bir sözlük oluşturacağız. Ama bu sefer ona biraz daha model ekleyeceğiz ve her modelin aykırı değerleri nasıl tahmin ettiğini göreceğiz.

Aykırı değer kısmının değerini probleminize ve verileri anladığınıza göre ayarlayabilirsiniz. Örneğimizde, verilerin geri kalanına benzemeyen% 5'lik gözlemleri tespit etmek istiyorum. Öyleyse, aykırı değer kesirinin değerini 0.05 olarak ayarlayacağım.

random\_state = np.random.RandomState(42)

outliers\_fraction = 0.05

# Define seven outlier detection tools to be compared

classifiers = {

'Angle-based Outlier Detector (ABOD)': ABOD(contamination=outliers\_fraction),

'Cluster-based Local Outlier Factor (CBLOF)':CBLOF(contamination=outliers\_fraction,check\_estimator=False, random\_state=random\_state),

'Feature Bagging':FeatureBagging(LOF(n\_neighbors=35),contamination=outliers\_fraction,check\_estimator=False,random\_state=random\_state),

'Histogram-base Outlier Detection (HBOS)': HBOS(contamination=outliers\_fraction),

'K Nearest Neighbors (KNN)': KNN(contamination=outliers\_fraction),

'Average KNN': KNN(method='mean',contamination=outliers\_fraction)

}