Değişken dönüşümleri

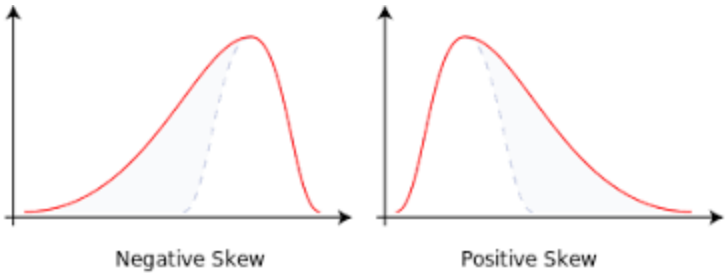
Bazı durumlarda değişkenleri normal dağılıma uygun şekilde dönüştürmek algoritmamızın daha iyi çalışmasını sağlayabilir.

Değişken dönüşümleri için bazı yollar vardır

1. Log
2. Log1p
3. boxcox

Bu yöntemler için gerekli kütüphaneler aşağıdaki kodlar ile içe aktarılabilir.

```
from numpy import log,log1p
from scipy.stats import boxcox
```

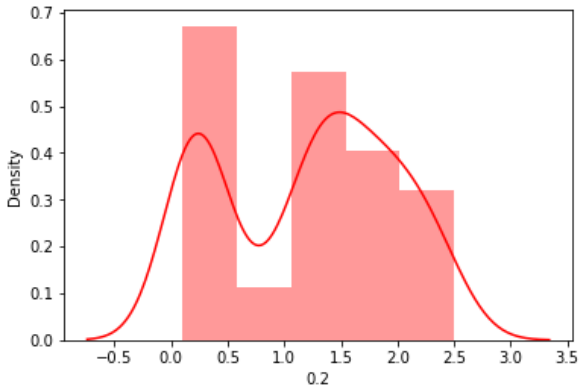


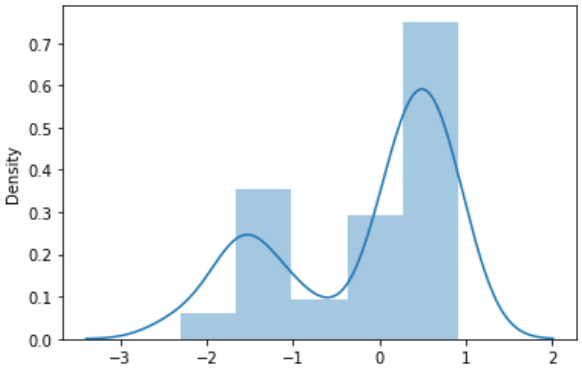
Pozitif ve negatif çarpıklığın görselleştirilmesi.

Logaritmik dönüşüm

```
import math
log_data = [math.log(d) for d in veri["kolon"]]
sns.displot(log_data)
```

Yukarıdaki kod kullanıldıktan sonra Aşağıda gösteriken ilk grafik verinin çarpık halini ikinci grafik ise logaritmik dönüşüm ile dönüştürülmüş halini gösterir.





Logaritmik dönüşüm Lineer Regresyon için kullanılabilir.

Sonuç olarak verideki değişkenlerin herbiri ya da seçilen biri logartimik olarak dönüştürülmüştür. Ancak lineer regresyon durumu bozulmamıştır.

Örneğin y = ax + b şeklindeki regresyon denklemi y = a(log(x)) + b şeklinde dönüştürülür.

Eğer verinizde 0 varsa log(0) tanımsız olduğu için bu yöntem çalışmayacaktır.

Polinomal değişkenler

Logaritmik yapabildiğimiz gibi verilerimizi yüksek dereceli şekilde polinomal olarakta değiştirebiliriz.

Yani bir özelliği dahil etmek yerine o özelliğin karesini küpünü ya da istediğniz derecesini dahil ederek veriye esneklik kazandırabiliriz.

Bu durumda düz bir çizgiyle değil bir eğriyle karşışalırız. Bu eğrinin derecesini kendimiz belirleyebiliriz.

Değişen derecesi arttırıldıkça eğrideki bükülmelerde artacaktır. Türev yani, 4.dereceden bir polinomun türevi 3.derecedendir. Yani 3 adet bükülme noktası bulunur.

Bunun için önce gerekli kütüphaneyi içeri alalım ve ondan bir nesne oluşturarak gerekli kodu yazalım.

```
from sklearn.preprocessing import polynomial_features
pf = polynomial_features(degree = n)
veri_poly = pf.fit_transform(veri)
```

Bu şekilde değişkenimiz n. dereceden değiştirilmiş oldu

Değişken Seçimi (Feature Selection)

Değişkenler seçilmeden önce değişkenlerde bazı işlemlerin yapılması gerekir Bunlar özetle şu şekildedir.

1. Değişken dönüşümü
2. Numerik olmayan veriler verilerin numeriğe dönüştürülmesi (Encoding)
3. Ölçeklendirme (Scaling)

Encoding (Kodlama)

Kategorik değişkenlerin kendi içerisinde iki tipi bulunur. Bunlar nominal ve ordinal veri tipleridir.

Kendi aralarında sıralanabilen fakat nicel olarak karşılaştırılamayan veriler **Ordinal** verilerdir.(Örneğin Mükemmel , Çok iyi , iyi , kötü , çok kötü)

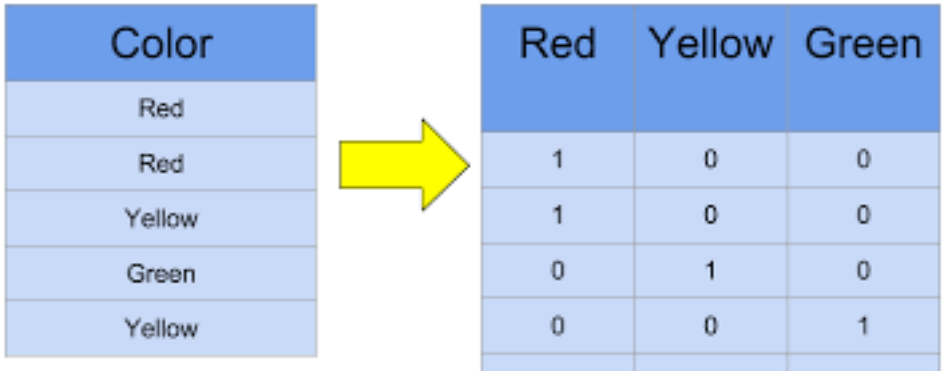
Kendi aralarında hem sıralanamayan hemde karşılaştırılamayan veriler **Nominaldir.** (Örneğim kırmızı yeşil mavi)

Encoding çeşitleri

1. **İkili kodlama (Binary Encoding)** - Mantıksal operatörleri (True (1) False (0)) kullanarak yapılan encoding çeşididir.Bir değişkende İki farklı tipte veri varsa kullanılabilir. Örneğin Evli-bekar, Sigara içiyor-Sigara içmiyor vs.

   ```
   veri = (veri == "aa").astype(int)
   #Değişendeki veri aa ise 1 Başka birşey ise 0 olarak dönüştürülür. İki tür veri olduğu
   #için veriler Binary Encodinge uygun olarak encode edilmiş olur.
   ```

2. **One Hot encoding** - Eğer değişkende birden fazla tipte veri bulunuyorsa kullanılabilecek bir encoding çeşididir. Her veri tipini bir sütüna ayırarak 0 ve 1ler ile ifade eder. Örneğin ;

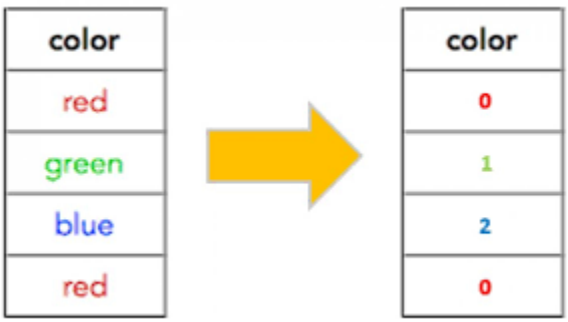


Yukarıdaki 3 farklı renk verisini üç farklı kolona atayarak 1ler ve 0 lar ile ifade edebiliriz. Renk kırmızı ise kırmızı kolonu 1 diğreleri 0 değerini alır.

Bunun için aşağıdaki iki kod kullanılabilir

```
from pandas import get_dummies
veri_Encode = get_dummies(veri,drop_first = True)
```

3. **Label Encoding** - LabelEncoder kullanıldığında veri seti aynı şekilde tek kolon halinde kalır. Fakat verilerin birbiri üzerinde üstünlük kurması durumu söz konusudur.



Şekilde görüldüğü üzere kımızı 0 yeşil 1 mavi 2 şeklinde kodlanır. Bu durumda verilerin birbiri üzerine üstünlüğü söz konusudur. Örneğin makine öğrenmesine verilen eğitim setinde mavi kırmızıdan üstün sayılacaktır.

4. **Ordinal Encoding -** Ordinal encoding Label encoder'a çok benzeyen bir encoding çeşididir ancak aralarında ufak bir fark bulunur.

Label encoder bir satırı encode ederken Ordinal encoder birden fazla satırı encode edebilir.

Öznitelik Ölçeklendirme (Feature Scaling)

Ölçeklendirmeyi değişkenleri aynı boyutta incelemek istediğimizde kullanırızç Bazı algoritmalar içinse ölçeklendirme bir şarttır. (Örn. SVM)

Örneğin evlerin metrekaresini belirten bir veriyi ele alalım. 90 ile 2000 arasında değişen veriler makine öğrenimi aldoritması için yararlı değildir çünkü verinin maksimum ile minimumu arasındaki uzaklık çok fazladır. Bu durumda ölçeklendirme kullanarak veriyi daha küçük iki sayı arasında ölçeklendiririz. Aynı şekilde veri 0.001 ve 0.002 arasında dağılıyorsa bu durumda verinin dağılım aralığı çok küçüktür. Yine scaling kullanarak veri daha uygun hale getirilir.

Ölçeklendirme çeşitleri

1. **Standard Scaling (Standardizasyon)** - Ortalama değerin 0, standart sapmanın ise 1 değerini aldığı, dağılımın normale yaklaştığı bir metoddur. Formülü şu şekildedir, elimizdeki değerden ortalama değeri çıkartıyoruz, sonrasında varyans değerine bölüyoruz.

   z = (x - u) / s

   Aşğıdaki kod ile kullanılabilir

   ```
   from sklearn.preprocessing import StandardScaler
   scaler = StandardScaler()
   veri_ölçekli = scaler.fit_transform(veri)
   ```

2. **Min-Max Scaling (Normalizasyon)** - Verinin minimum değerinin 0, Maksimum değerinin 1 ile değiştirilerek ölçeklendirildiği durumdur. Burada ‘outlier’ denilen dışta kalan verilere karşı hassasiyet durumu vardır, bu yüzden bu değerlerin fazla olduğu bir durumda iyi bir performans gösteremeyebilir.

   Aşağıdaki kod kullanılabilir.

   z = x - min(x) / (max(x) - min(x)

   ```
   from sklearn.preprocessing import MinMaxScaler
   Scaler = MinMaxScaler()
   ```

```
veri_ölçekli = Scaler.fit_transform(veri)
```

3. **Robust Scaling** - Normalizasyon ile benzer şekilde çalışır. Aykırı değerlere sahip verilerde daha iyi sonuçlar verebilir. Yine veri dağılımı ile benzerlik gösterir ancak aykırı değerler dışarıda kalır. Medyan değeri sonradan kullanılmak üzere elenir ve değerler 1.ve 3. kartil aralığına oturtulur.

   z = x - Q1(x) / Q3(x) - Q1(x)

```
from sklearn.preprocessing import RobustScaler
Scaler = RobustScaler()
veri_ölçekli = Scaler.fit_transform(veri)
```

4. **MaxAbs Scaling (Maksimum mutlak değer ölçeklendirmesi) - H**er özelliğin maksimum mutlak değeri 1 olacak şekilde her özelliği ayrı ayrı ölçeklendirir ve dönüştürülür.

   z = x / max(abs(x))

```
from sklearn.preprocessing import MaxAbsScaler
Scaler = MaxAbsScaler()
veri_ölçekli = Scaler.fit_transform(veri)
```

5. **PowerTransformer -** Varyansı stabilize etmek ve çarpıklığı en aza indirmek için en uygun ölçeklendirme faktörünü bulur. Yine ortalama değerin 0, standart sapmanın ise 1 değerini aldığı bir metoddur.

```
from sklearn.preprocessing import PowerTransformer
Scaler = PowerTransformer()
veri_ölçekli = Scaler.fit_transform(veri)
```

Mert Aydoğan Gazi Universiy