**Understanding Missing Values in Feature Engineering**

When working with real-world data, missing values are a common problem. In feature engineering, handling missing values correctly is crucial because they can affect the performance of machine learning models.

**Types of Missing Values**

Missing values can be categorized into three main types:

**1. Missing Completely at Random (MCAR)**

* The missing values occur randomly and have no relationship with any other variable in the dataset.
* Example: A person forgets to fill in their age in a survey.
* **Effect:** No bias is introduced, and dropping missing values or imputing them with a simple method like mean/median is usually fine.

**2. Missing at Random (MAR)**

* The missing values are related to other observed variables but not to the missing data itself.
* Example: In a medical dataset, men are more likely to skip answering a question about weight, but within each gender, weight is missing randomly.
* **Effect:** More advanced imputation methods like regression or using related features to fill in the missing values can be useful.

**3. Missing Not at Random (MNAR)**

* The missing values depend on the unobserved data itself, meaning the reason for missingness is not random.
* Example: People with higher incomes may choose not to report their salaries in a survey.
* **Effect:** This can introduce bias, and handling MNAR often requires domain knowledge or special techniques like modeling the missingness separately.

**Handling Missing Values in Feature Engineering**

Once we identify the type of missing values, we can decide how to handle them:

**1. Removing Missing Values (If Missing is Low)**

* If a feature has very few missing values (e.g., <5% of the dataset), simply removing those rows may not impact the model significantly.
* Example:

df.dropna(inplace**=True**)

**2. Imputation (Filling Missing Values)**

* **Mean/Median/Mode Imputation**: Replacing missing values with the mean (for numerical data), median (for skewed numerical data), or mode (for categorical data).

df['Age'].fillna(df['Age'].mean(), inplace**=True**) *# Mean Imputation*

* **Forward or Backward Fill**: Filling missing values using previous or next available values (good for time series data).

df.fillna(method**=**'ffill', inplace**=True**) *# Forward fill*

* **Predictive Imputation**: Using models like regression or KNN to estimate missing values.
* **from** sklearn.impute **import** KNNImputer
* imputer **=** KNNImputer(n\_neighbors**=**3)

df[['Age', 'Salary']] **=** imputer.fit\_transform(df[['Age', 'Salary']])

**3. Adding Missing Indicators**

* Instead of filling missing values, we can create a new column that flags whether a value was missing.

df['Age\_missing'] **=** df['Age'].isnull().astype(int)

**4. Using Domain Knowledge**

* In cases of MNAR, sometimes domain expertise can help us decide on the best approach.
* Example: If missing income values indicate high salaries, we might group them as "high income" instead of just imputing with the mean.

**Summary**

| **Type of Missing Data** | **Example** | **Best Handling Methods** |
| --- | --- | --- |
| **MCAR** | Randomly missing age values in a survey | Drop rows or use simple imputation |
| **MAR** | Missing income values depend on education level | Use regression or related features for imputation |
| **MNAR** | High earners not reporting salaries | Use domain knowledge or model missingness |

**Final Tip:**

Before deciding how to handle missing values, always **explore your data** to understand the reason behind the missingness. Proper handling can significantly improve your machine learning model's accuracy!

🡪feature Enginnering’te amaç eksik veya yanlış verileri düzeltmeye denir. Üçe ayrılır;

**1. Missing Completely at Random (MCAR)**

🡪Random eksik veriler dataset’te birbirleriyle alakası olmayan ve herhangi bir mantık olmayan veri eksikliğidir.

**2. Missing at Random (MAR)**

🡪Burada ise anlam şu eksik veriler diğerleriyle ilişkili ama kendisiyle değil gibi bir anlam var o da şu.

🡪 Bir hastaneden erkekler kilolalarını belirtmeyi sevmiyorlar ve girmiyorlar.

**3. Missing Not at Random (MNAR)**

**🡪**Burada ise eksik veriler ….

**Handling Missing Values in Feature Engineering**

Birçok farklı yolla ise düzenlemeyi yapabiliyoruz eksik verilerimiz üzerinden.

**1. Removing Missing Values (If Missing is Low)**

Ex:df.dropna(inplace=True)🡪dropna ile verileri silebiliyoruz.

**2. Imputation (Filling Missing Values)**

🡪df['Age'].fillna(df['Age'].mean(), inplace**=True**)*# Mean Imputation*

**🡪**İlk etapta, titanic verimi çekiyorum.

metin, ekran görüntüsü, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Hangi verilerin eksik olduğunu buradan görebiliyorum.

Sonra “box-plot” ile median ve mean hangisini boş veriler yerine yazmak daha mantıklı olur ona bakmak için “bax-plot” çağırıyorum.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Su şekilde eklemeleri yapıyorum.

Daha sonra “embarked” kısmında iki tane boş yer var onu doldurmak için mode’a göre alıyorum çünkü mantıkken veri yüksek ve iki tane tek eksik verimiz var . Bu nedenle en çok tekrar edeni oraya yazmak daha mantıklı.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, çizgi, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Bu şekilde ayarlanmış olmakta.

BALANCING DATA

Burada ise amacımız düzensiz verileri düzeltmek.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk etapta böyle random bir veri seti oluşturuyoruz. Amacımız burada yeni gelen bilginin hangisine ait olacağını bilmek. Daha makine öğrenmesi yapmadığımız için tahmin ve benzeri bir şey yapamıyoruz. Verilerimizde düzensiz biri 900 tane “0” diğeri 100 tane “1”.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Şimdi verimize yeni bir veri geldiğinde doğru şekilde sınıflandırmamız gerekiyor gelen veri doğru şekilde sınıflandırıldı mı bunu bilmek gerekiyor. Şimdi verimize bakınca 900 tane “0” ,100 tane “1” var gelen yeni veri doğru şekilde “target” alabilecek mi;

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Dengesiz data’yı bu şekilde iki farklı yolla daha dengeli yapabilir ve daha doğru sonuçlar almaya yardım edebiliriz. Burada asıl amaç makine öğrenmesi için hangi yolların daha iyi sonuç vereceğini öğrenmek.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk etapta “upsamling” yapacağım yani 100 tane “1” vardı sadece bunu 900’e çıkartıp daha dengeli bir veri seti elde etmeye çalışacağım bunu yaparken de makine öğrenmesinde çok işimize yarayan bir kütüphaneden yaralanıyoruz;

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Burada ilk etapta “sklearn” kütüphanemizden “resample” fonksiyonumuzu import ediyoruz. İlk etapta hangi veriyi düzenleyeceğini giriyoruz,”replace” default olarak True geliyor. Daha sonra “n\_samples” ile hangi dataset’teki veri kadar olacaksa onun adını giriyoruz. Daha sonra da default olarak “random\_state=42” yapıyoruz Yukarıda random olarak oluşturduğumuz sayıları almış olduk Yani artık ne zaman “42”yi çağırsak ilk oluşturduğu random sayıları verecek. Ve bize 900 tane “1” dönmüş oluyor.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Daha sonra birleştiriyorum bunları ve eşit sayıda “1” ve “0” var artık.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Downsample’da aynı şekilde yapılmakda.

UPSAMPLE ÇOK ÖNEMLİ ÖRNEK

Titanic Dataset’imizi alıyoruz  
metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Burada “deck” kolonumuz ‘da 688 tane eksik var ve ben bunları tek tek manuel vs girmek yerine “umsample” ile belirli bir mantığa göre yapıstımak istiyorum.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

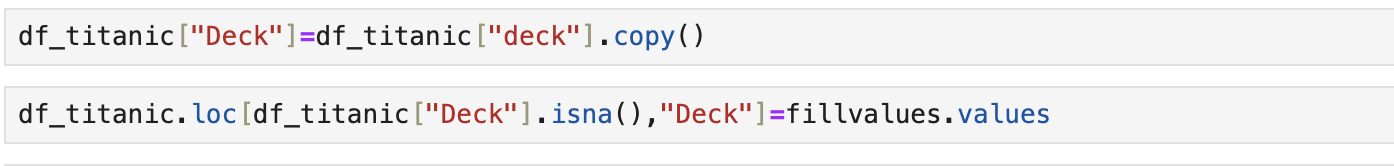
Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk etapta kütüphanelerimi import ediyorum. Daha sonra “non\_mising” parametresini oluşturup burada “titanic[“deck”]” içerisindeki boş verileri siliyorum.



Burada “resample” fonksiyonum içerisine “non\_missing” değerimi alıyorum. Çünkü non\_missing içinde non olmayan değerler var “ resample” fonksiyonunda ilk verdiğimiz parametredeki değerlere göre boş değerlere ekleme yaptığı için “NaN” olanları alırsak “NaN” değerlere yine “NaN” atayabilir bu da saçma olur. “n\_samples” bize kaç tane o değerden oluşturacağını belirtiyor biz “deck” içerinde “NaN” olan kadar oluşturacağız ki bizdeki boş yerlere yazsın. Eğer tüm sütun uzunluğu kadar üretirsek (len(df\_titanic)), bu sefer **tamamen yeni bir kolon** olur, eksik olmayanlar da değişir.

En sonda “reset\_index” dememizin sebebi ise bizim “non\_missing” içerisindeki indexler silinen veriler de olduğu için karısık “(1,3,5,11)” vs bu nedenle en baştan alsın diye.



Daha sonra “df\_titanic[“Deck”]” adında yeni sütünumuzu oluşturuyoruz ve normal “deck”’i kopyalıyoruz. “df\_titanic.loc[df\_titanic[“Deck”].isna()” direkt boş olan kısımları bulur ve “fillvalues.values” ile sırayla değerleri yazıyoruz. Ve değerler eklenmiş oluyor.

metin, sayı, numara, yazılım, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Mesela aynı şekilde bu örnek üzerinden gidelim.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk etapta eksik verileri kaldırıp bir değiştkene atıyorum. metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Sonra bu şekil oluşturuyorum ve totalde 3 eksik verim olduğu için bana 3 veri oluşturdu.

metin, sayı, numara, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Eğer “n\_samples”’ı “Paris”’teki eksik veri kadar oluştur demek yerine “df\_weather” uzunluğu kadar oluştur deseydim . Baştan sona verilerimiz de değişmiş olacaktı.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Onun için uzunluğu kadar oluşturmadım . daha sonra bu şekild eyeni bir kolon oluşturup “Paris”’e kopyaladım çünkü eksik verilere göre yeni veri ürettik 3 yeni veriyi 3 eksik satıra ekleyeceğiz. Daha sonra  
metin, sayı, numara, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

“loc” ile boş olan yerleri bulup sırayla ekletmiş olduk. Hem verimiz değişmemiş oldu hem de belli bir mantıkta yapmış olduk.

SMOTE

Smote’e ise bize şunu sağlamakta “upsample”’da verileri çoğaltırken yaptığı şey verileri kopyalamak gibi 100’den 900’e çıkartırken yaptığı şey 9 defa 100’ü yazdırmak gibi bu bazen mantıklı sonuç vermemekte bunun için “SMOTE” kullanılmakta.

ekran görüntüsü, metin, renklilik içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk etapta “feature\_1” ve “feature\_2” değerlerimizi gösteriyoruz. Daha net anlamak için grafik kullanıyoruz.

metin, yazı tipi, çizgi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Daha sonra bu şekilde kütüphanemizi dahil ediyoruz sonra “Smote()”’a bir değer atıyoruz. Daha sonra “fit.resample()” kütüphanesi ile “y=target” ve “x= feature”’larımızı ekliyoruz. Bu şekilde hem yeni “feature”’ler hem de yeni “target”’lar oluştu ayrı ayrı oluşmakta burada . “Concat” ile birleştiriyoruz.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Bu şekilde birleştiriyoruz .  
ekran görüntüsü, metin, renklilik, diyagram içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Tekrar grafiği çizince farkı bu sefer net şekilde görüyoruz. Mantık çerçevesinde değerlerle düzenledi veriyi. Burada net şekilde görüyoruz ki direkt eski sarı noktaların üzerine eklenmemiş yeni sarı noktalar elde edilmiş Sadece “upsample” kullansaydık aynı noktalar yenilenecekti

ENCODING

Encoding mantığı tam olarak şu “Kategorik “ bir değeri “Numeric” bir değere çevirmekte kullanıyoruz. Mesela “red” yerine “1” vermek gibi daha kolay şekilde makine anlasın diye bunu yapıyoruz. Yaklaşık 6 tane var;

Tekrar “titanic” dataset’imizi alıyoruz . Eksik verilere ekliyorum ya da çıkarıyorum

metin, ekran görüntüsü, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Bu şekil ayarlıyoruz.

Encoding çeşitlerimizden ilk olan one-hot encoding;

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Burada ki mantık şu “cinsiyet “ değerleri kadın, erkek diye ayrılmış bu şekilde kategorize edilmiş bunu encoding ile 1 ve 0 ‘ a çeviriyoruz. İlk etapta “pandas’ta” buna benzeyen bir fonksiyon var onunla yapıyoruz.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Değerlerimiz bu şekilde hangi grupta ne olduğunu görebiliyoruz.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

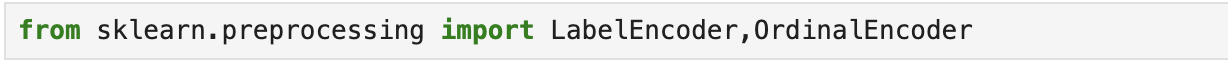
Pandas’ta ama bu şekilde saçma yapıyor direkt “FALSE “ veya “TRUE” olarak döndürüyor. Bunun için farklı yol ile yaapcağız.

LABEL ENCODING;

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Bu da encodig’in farklı bir versiyonu burada değerler otomatik olarak veriliyor.



metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk olarak ilk df’imizi kopyalıyoruz. Daha sonra “LABEL\_ENCODER” fonksiyonlarımızı kullanabilmek için “LabelEncoder()” fonksiyonumuzu bir değişkene atıyoruz. Bu şekilde atadıktan sonra.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Daha öncede “upsample”’da kullandığımız gibi “fit.transform” fonksiyonumuzu kullanıyoruz. Ve sonuç bir ve sıfırlardan oluşmuş oldu.

ORDINAL ENCODING;

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Ordinal Encoding’te ise mantık “LABEL” ‘a benziyor ama burada manuel olarak sınıflandırıyoruz.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

İlk etapta kopyalıyoruz. Daha sonra kategori kısmı için “DF” imiz içerisinde yazan şekilde sınıflarımızı yazıyoruz. Daha sonra “ORDINAL ENCODER” fonksiyonumuzu bir değere atıyoruz daha sonra . metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

“df\_order” ımız içerisindeki sınıfımızı bu şekilde değiştiriyoruz.

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Bu şekilde de istersek hepsini grafiklerde gösterebiliyoruz.