FEATURE ENGINEERING

What is feature engineering?

* Creation of new featured based on existing features
* Insight into relationships between features
* Extract and expand data
* Dataset-dependent

Feature engineering is the process of using domain knowledge to extract new variables from raw data that make machine learning algorithms work.

For most other use cases companies face, feature engineering is necessary to convert data into a machine learning-ready format. The choice of features is crucial for both interpretability and performance. Without feature engineering, we wouldn’t have the accurate machine learning systems deployed by major companies today.

**1- Encoding (categorical variables)**

* Encoding binary variables
  + Pandas
  + Scikit-learn (label encoder)
* One hot encoding
  + Pandas (get\_dummies())
  + Scikit-learn (OneHotEncoder)
* Mean encoding

**2- Engineering numerical features**

* Aggregate statistics (etc. “mean encoding”- target variable’deki toplam orani)
* Dates

**3- Engineering features from text**

* Extraction
* Vectorizing text
* Text classification

1- **Encoding (categorical variables)**

**• Encoding binary variables**

“yes/no” gibi sadece iki deger alan bir degiskenin “1/0” olarak donusturulmesi

Pandas:

In [2]: users["sub\_enc"] = users["subscribed"].apply(lambda val: 1 if val == "y" else 0)

Scikit-learn (LabelEncoder)

In [4]: from sklearn.preprocessing import LabelEncoder

In [5]: le = LabelEncoder()

In [6]: users["sub\_enc\_le"] = le.fit\_transform(users["subscribed"])

# her iki islemin sonucu da birebir aynidir, “0” ve “1”lerden olusan 1 tane degisken (feature/column) elde etmis oluruz (Bu nokta onemli, kategori sayisi kadar degil, 1 tane feature var yine elimde)

**• One hot encoding**

Ikiden fazla kategoriler icin ilk adim yetersiz kalir, cunku kategori icin verilen numaralari algoritmalar buyukluk ve kucukluk olcusu olarak kullanarak, kategoriler arasinda, aslinda var olmayan, niceliksel bir iliski kurar. Bunu onlemek icin her bir kategori icin bir sutun olusturmak gerekir.

Pandas:

pd.get\_dummies(users["fav\_color"])

# bu metot uygulandiginda, kategori sayisi kadar yeni “feature” elde edilmis olur.

Scikit-learn (OneHotEncoder)

from sklearn.preprocessing import OneHotEncoder

onehotencoder = OneHotEncoder(categorical\_features = [0])

x = onehotencoder.fit\_transform(x).toarray()

# tahditleri:

Bu kod sadece 0 ve 0’dan buyuk degiskenler uzerine uygulanabilir. Yani bundan farkli bir durumda once LabelEncoder veya onun islevini gorecek bir metot uygulanmalidir.

Regularization’i azaltmak icin dummy variable’lerden birinin cikarilmasi gerekir. Ancak Dummy feature’lar ayni degiskene atandagi icin (bundan emin degilim), bir kategoriye dusurmek mumkun olmaz

Hangi dummy’i attigi anlasilamayabilir (adam burada ikinci tespiti ile celismis zaten, hemen cikaramiyoruz diyor, hemen sonra da hangisini cikardigimizi anlayamayiz diyor)

* Mean encoding

High Cardinality : When number of unique elements of a column is significanltly high.

Burada ilgili degerin, target’da 1 oldugu durumlarin toplam gozlem sayisina bolunmesi ile elde ediliyor. Mesela, “bora” ismi 5 kere gorunmus ve bunlardan ikisinde target variable “1” olmus olsun. O zaman “2/5=0.4” olacak ve “bora” yerine “0.4” yazicaz. Peki target variable binary degil ise ne olacak, onu henuz ogrenmedim☺

**2- Engineering numerical features**

Aggregate statistics

In [2]: columns = ["day1", "day2", "day3"]

In [3]: df["mean"] = df.apply(lambda row: row[columns].mean(), axis=1)

# burada ayni sehrin 3 gunu iceren sicaklik bilgilerinin ortalamasi alinmis. Yani 3 yerine 1 tane degisken kullanilacak modelde ve cok boyutluluk da azaltilmis olacak boylece.

Dates

In [6]: df["date\_converted"] = pd.to\_datetime(df["date"])

In [7]: df["month"] = df["date\_converted"].apply(lambda row: row.month)

# Burada da tarih bilgisinden, sadece ay bilgisi cekilerek yeni bir feature olusturulmus.

**3- Engineering features from text**

Extraction

In [1]: import re

In [2]: my\_string = "temperature: 75.6 F"

In [3]: pattern = re.compile("\d+\.\d+")

In [4]: temp = re.match(pattern, my\_string)

In [5]: print(float(temp.group(0))

75.6

# burada “float” bir datadan istedigim bilgiyi string olarak cektim. \d+, integer’larin alinmasini, “\.”, decimal noktasinin, ikinci d+ ise decimalden sonraki numaralarin alinmasini sagliyor.

Vectorizing text

tf = term frequency

idf = inverse document frequency

In [6]: from sklearn.feature\_extraction.text import TfidfVectorizer

In [8]: tfidf\_vec = TfidfVectorizer()

In [9]: text\_tfidf = tfidf\_vec.fit\_transform(documents)

Text classification

P (A∣B) = P (B∣A)P (A)

P (B)

That experience part is how you use the data to make informative features for the model to learn better. That’s Feature Engineering.