Talk with the dataset and understand what is it saying.....

Day 1: Zomato Dataset

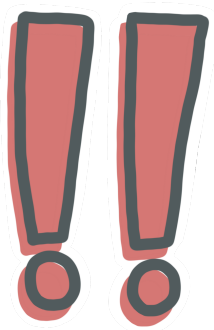
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline
```

} importing the necessary lib

```
df = pd.read_csv('zomato.csv')
```

Error !!!

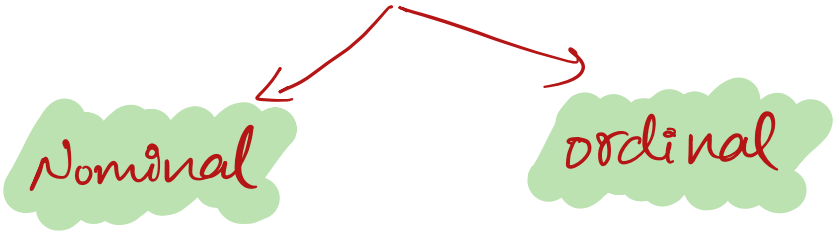
Encoding:



→ a ML algo should understand the data it gets.
→ for example, "small", "medium", "large" need to be converted into numbers.
→ to solve, we given them small-1, med-2, large-3 } for example

But, this is just an example and it's not the best way.....

Encoding is of mainly two types:



1. one hot / dummy encoding
2. Label / ordinal encoding
3. Target encoding
4. Frequency / Count encoding
5. Binary encoding
6. Feature Hashing

all types

we take an example dataset :—

	player	point
0	Stephen Curry	4
1	Anthony Edwards	6
2	Anthony Edwards	1
3	Duncan Robinson	2
4	Duncan Robinson	3
5	Stephen Curry	5
6	Stephen Curry	2
7	Stephen Curry	3

Before moving further, we will understand few terms :—

(1) Nominal Variables - are variables that have no inherent order. They are simply categories that can be distinguished from each other...

(2) ordinal variables - have an inherent order. They can be ranked from heighest to lowest (or vice versa.

(3) Unsupervised encoding methods - don't make use of the target variable to encode categorical variables.
(eg encode player names with a formula that takes the no. of points that they made)

(4) Supervised encoding variables - employ the target variable to encode categorical variables.

(5) The "Cardinality" of a Categorical Variable - stands for the number of categories represented by this variable

6. **Target leakage**: occurs when a variable is used for training but would not be available at inference time.

One-Hot / Dummy Encoding:

→ In this, categorical data are represented as vectors of zero and one.

	player_1	player_2	player_3	point
0	1	0	0	4
1	0	1	0	6
2	0	1	0	1
3	0	0	1	2
4	0	0	1	3
5	1	0	0	5
6	1	0	0	2
7	1	0	0	3

```
from category_encoder import OneHotEncoder
OneHotEncoder(cols=['player']).fit(df).transform(df)
```

- `OneHotEncoder` — function
- `cols=['player']` — what column?
- `.fit(df)` — fit the dataset
- `.transform(df)` — transforming the present to this form.

Label / Ordinal Encoding:

→ probably the simplest way
→ The categorical data is converted into numerical data.
⇒ each category is assigned by a numerical value.

```
from category_encoder import OrdinalEncoder

mapping=[{'col':'player', 'mapping':
        {"Stephen Curry":1,
         "Duncan Robinson":2,
         "Anthony Edwards":3}}]

OrdinalEncoder(cols=['player'],
mapping=mapping).fit(df).transform(df)
```

note: Just like one hot encoder.

O P

⇓

	player	point
0	1	4
1	3	6
2	3	1
3	2	2
4	2	3
5	1	5
6	1	2
7	1	3

Target Encoding:

→ we encode using a target variable.

→ if the feature is a good predictor of the target, then its values should be closer to the target

① Target-Mean Encoding:

→ we replace the category with the mean of the target values.

→ This method is used to avoid target leakage

② Leave-one-out Encoding:

→ similar to above, but we leave the sample which we predict and take rest targets....

```
from category_encoder import TargetEncoder
TargetEncoder(cols=['player'], smoothing=1.0).
    fit(df, df['point']).transform(df)
```

```
from category_encoder import LeaveOneOutEncoder
LeaveOneOutEncoder(cols=['player']).
    fit(df, df['point']).transform(df)
```

	player	point
0	3.5	4
1	3.5	6
2	3.5	1
3	2.5	2
4	2.5	3
5	3.5	5
6	3.5	2
7	3.5	3

Frequency / Count Encoding:

→ represent data using the count of the categories

→ Frequency encoding is simply a normalized

Version of Count encoding

```
from category_encoder import CountEncoder

CountEncoder(cols=['player']).fit(df).
                              transform(df)
```

	player	point
0	4	4
1	2	6
2	2	1
3	2	2
4	2	3
5	4	5
6	4	2
7	4	3

```
from category_encoder import CountEncoder

CountEncoder(cols=['player'], normalize=True).
                    fit(df).transform(df)
```

	player	point
0	0.50	4
1	0.25	6
2	0.25	1
3	0.25	2
4	0.25	3
5	0.50	5
6	0.50	2
7	0.50	3

Binary Encoding:

→ transform cat.. data into numerical data by encoding categories as integers and then converting them into binary code.

```
from category_encoder import BinaryEncoder

BinaryEncoder(cols=['player']).fit(df).
                                transform(df)
```

	player_0	player_1	point
0	0	1	4
1	1	0	6
2	1	0	1
3	1	1	2
4	1	1	3
5	0	1	5
6	0	1	2
7	0	1	3

Feature hashing:

→ Representing data in high-dimensional space using a fixed-size array.

→ with the help of a hash function.

```
from category_encoder import HashingEncoder
HashingEncoder(Cols=['player']).fit(df).
                                transform(df)
```

	col_0	col_1	col_2	col_3	col_4	col_5	col_6	col_7	point
0	0	0	0	0	1	0	0	0	4
1	0	0	1	0	0	0	0	0	6
2	0	0	1	0	0	0	0	0	1
3	0	0	1	0	0	0	0	0	2
4	0	0	1	0	0	0	0	0	3
5	0	0	0	0	1	0	0	0	5
6	0	0	0	0	1	0	0	0	2
7	0	0	0	0	1	0	0	0	3

now, continuing our zomato dataset :-

→ `df = pd.read_csv('zomato.csv', encoding='latin-1')`

→ `df.head()`

→ `df.columns()`

→ `df.info()`

→ `df.describe()`

Steps:

DON'T FORGET

1. Missing values
2. Explore about the numerical variables
3. Explore about categorical variables
4. Finding Relationship between features

→ df.isnull().sum()

→ [features for features in df.columns if df[features].isnull().sum() > 0]

→ sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')

→ pd.merge(df, df_country, on='Country Code', how='left')

→ final_df.dtypes()
final_df.value_counts()
final_df.value_counts().index
final_df.value_counts().values

Sample:

→ a randomly chosen from population.
→ we calculate covariance and correlation on samples rather than the complete population.

Covariance:

→ it is dependent only on sign
→ it is a measure used to denote how much variable change randomly.
→ lies b/w $-\infty$ to $+\infty$

For population

$$Cov(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y}_i)}{N}$$

For Sample

$$Cov(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{(N-1)}$$

where,

$\bar{X}$ = Sample mean of x

$\bar{Y}$ = Sample mean of Y

X_i and Y_i = the values of X and Y for the ith record in the sample.

N = no. of records in the sample.

Correlations: (pearson's correlation)

↳ pear

→ lies b/w −1 to 1

$$\text{Correlation} = \frac{Cov(x,y)}{\sigma(x) * \sigma(y)}$$

→ $Cov(x,y)$ = Sample covariance b/w X and Y

σx = Sample standard deviation of X

σy = " " " " Y

Spearman's correlation Co-efficient:

$$= \frac{\text{Covariance}(\text{rank}(x), \text{rank}(y))}{(\text{stdv}(\text{rank}(x)) * (\text{stdv}(\text{rank}(y)))}$$

→ lies b/w −1 to 1

→ spearmanr()