5/4/22
10:30pm

3 * Discretization *

Converting Continous Data To Discrete Data

also called as "Binning"

Code :-

```
Import pandas as pd

# stroke = pd.read_csv ("stroke prediction.csv")

# stroke.head ()
```

Out

id	gend	age	hyper Tension	heart_disease	Ever_married	Work type	Residence Type	avg_glocose Level	bmi	Smok status	Stroke
30669	Male	3.0	0	0	No	children	Rural	95.12	18.0	NaN	0
30468	Male	58.0	1	0	Yes	Private	urban	87.96	39.2	Never	0
16523	Female	8.0	0	0	No	Private	urban	110.89	17.6	NAN	0
56543	Female	70.0	0	0	Yes	Private	Rural	69.04	35.9	Formerly smoked	0
46136	male	14.0	0	0	No	Never_Worked	Rural	161.28	19.1	NAN	0
	discrete (cat)	continous	discrete (cnst)	discrete (Count)	discrete	discrete	discrete	continous	contin	discrete	discrete

```
# stroke.shape
```

Out : (43400,12)

```
# stroke.info ()
```

Out   information about stroke data.

⇒ Create Bins

```
# intervals = [0,12, 19, 30, 60, 90] → # bins ⇒ Each interval

# categories = ["child", "teenager", "young adult", "middle aged",
                 "senior_citizen"]
```

The Age column is Continous Data. so, We converting into Discrete Variable

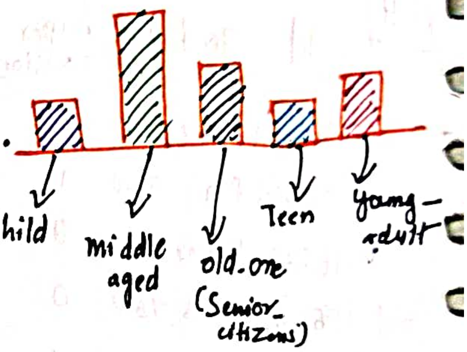
intervals	categorised
* 0-12 →	child
* 12-19 →	teenager
* 19-30 →	young adult
* 30-60 →	Middle aged
* 60-90 →	Senior citizen

* Creating New column & storing The data of "Age

```
# Storke ["Age_category"] = pd.cut [X =
        X = Stroke ['Age'], bins = intervals,
        Labels = categories]
```

Divide.

Storke ["Age category"]. hist ()



```
# stroke.head ()
```

Changed Continous Data To Discrete Categorical data

Newly-added

Out

Id	gender	age	hyper Tension	Heart-disease	Ever married	work type	Residence Type	Avg. glucose Level	bmi	Smoke	Stroke	Age-Categeory
30669	Male	3.0	-	-	-	-	-	-	-	-	-	child
30468	Male	58.0	-	-	-	-	-	-	-	-	-	middle-aged
16523	Female	8.0	-	-	-	-	-	-	-	-	-	child
56543	Female	70.0	-	-	-	-	-	-	-	-	-	Senior-citizen
46143	Male	14.0	-	-	-	-	-	-	-	-	-	teen

4 * Encoding *

Applicable For Discrete Categorical Variables

* Machine Can't Understand Text Data. So, We Convert the "discrete categorical" to "discrete count" Data

* There are two types of Categorical Data

1. Nominal (No Natural Sequence)
2. Ordinal (Natural Sequence)

we should Treat Every variable Equally.

Ex:- Grades

O	10
A+	9
A	8
B+	7
B	6
F	0

⇒ O > A+
10 > 9

odinal Data

cityname	
Hyd	0
Delhi	1
Mumbai	2
Kerala	3
Gujarat	4

In this case, we can't Decide by numbering given To state.

which is greater

⇒ To convert categorical Data To Numeric :

Nominal Data → (dummies)

1. get Dummies (pandas)
2. One hot Encoding (sklearn)

Odinal Data

1. map (pandas)
2. Label Encoder (sklearn)

# One Hot Encoding	# Label Encoding
* Nominal	* Odinal

```
import Pandas as pd
import Numpy as np
import matplotlib.pyplot as plt
% matplotlib inline
```

```
# df = pd.read_csv("homeprices.csv")
# df.head()
```

Out :-

	Town	area (sqft)	Price
0	chennai	2600	5500000
1	chennai	3000	5650000
2	chennai	3200	6100000
3	chennai	3600	6800000
4	Banglore	2600	5850000

Town

1. Chennai
2. Banglore
3. hydarabad

Nominal Data

Discrete → Categorical Data

⇒ **Pd.get_dummies** [Nominal Variable Encoding using pandas]

```
# dummies = pd.get_dummies (df ["Town"])
# dummies
```

Out :-

	Banglore	Chennai	Hyderabad
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	0	0	1
9	0	0	1
10	0	0	1

Out :-

	Banglore	Chennai	Hyderabad
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	0	0	1
9	0	0	1
10	0	0	1
11	0	0	1

→ Dummies

which Ever Record is there it shows = 1

* For other columns it shows = 0

Adding   original   converted one [by]

df_dummies = pd.Concat ([df, dummies], axis = "columns"

df_dumies

Out :-

	Town	Area	Price	Banglore	Chennai	Hyderabad
0	Chennai	2600	5500000	0	1	0
1	chennai	3000	5650000	0	1	0
2	Chennai	3200	6100000	0	1	0
3	chennai	3600	6800000	0	1	0
4	Banglore	3600	5850000	1	0	0
5	Banglore	2600	6150000	1	0	0
6	Banglore	2800	6500000	1	0	0
7	Banglore	3300	610000	1	0	0
8	Hyderabad	3600	7100000	0	0	1
9	Hyderabad	2600	5750000	0	0	1
10	Hyderabad	2900	6000000	0	0	1
11	Hyderabad	3100	6200000	0	0	1

Delete The Orignal column

```
#df_dummies . drop ("town", axis = "columns", inplace =True)

df_dummies
```

Out :

	Area	Price	Banglore	chennai	Hyderabad
0	2600	5500000	0	1	0
1	3000	5650000	0	1	0
2	3200	6100000	0	1	0
3	3800	6800000	0	1	0
4	2600	5850000	1	0	0
5	2800	6150000	1	0	0
6	3300	7100000	1	0	0
7	3600	5750000	1	0	0
8	2600	6000000	0	0	1
9	2900	6200000	0	0	1
10	3100	6900000	0	0	1
11	3600	6500000	0	0	1

after deleting orignal column in New Data (ie. df-dummies) Old_data (df) is still remains same.

∅ Dummy Variable Trap

⇒

Bang	chenni	Hyder
0	1	0
0	1	0
1	0	0
1	0	0
0	0	1
0	0	1

→

Bang	chenn
0	1
0	1
1	0
1	0
0	0
0	0

→ hyderabad

If we Remove One of The column also, still we can Identify, col name by (0,0)

Town — 3 caterogies

↓ Dummies

3 columms [Multi-collinearty Problem / colleration]

If column → "n" categories

↓ we convert

(n-1) column

df_dummies . drop ("chennai", axis="columns", inplace=True

df_dummies

out

	Area	Price	Banglore	Hyderabad
0	2600	5500000	0	0
1	—	—	0	0
2	—	—	0	0
3	—	—	0	0
4	—	—	0	0
5	—	—	1	0
6	—	—	1	0
7	—	—	1	0
8	—	—	1	0
9	—	—	0	1
10	—	—	0	1
11	36 000	6500000	0	1

(← Same) as old data

where ever Both column Having Equal Zero's, it i. another column's data.

Chennai = 0 0

We Can Write One line of Code To do all these

```
# df_dum = pd.get_dummies (df, drop_first = True)

# df_dum
```

Out :

	Area	Price	town_chennai	town_hyderabad
0 1 2 . . . 10 11			1 1 1 1	0 0 0 0

This Diagram is Exact of (Previous) Last page diagram. But deleted banglore Replaced chennai"

town_banglore [deleted] → Because, B ✓ Chennai ← hyd (Alphabet order

Replaced with (0 0)

⇒ One Hot Encoding → Nominal using sklearn

```
From sklearn.Preprocessing import One Hot Encoder

# enc = OneHotEncoder (drop = "First")
```

stores in enc

```
# enc_df = pd.DataFrame (enc.fit_transform
                (df [["town"]]).
                to array ()
```

converting

output

```
# enc-df
```

Out

	0	1
0	1.0	0.0
1	1.0	0.0
2	1.0	0.0
3	1.0	0.0
4	0.0	0.0
5	0.0	0.0
6	0.0	0.0
7	0.0	0.0
8	0.0	1.0
9	0.0	1.0
10	0.0	1.0
11	0.0	1.0

```
# merge with main df
# df_ohe =   df.join (enc-df)
# df-ohe.drop ("town", axis = "columns", inplace = True)

# df-ohe
```

Out

	area	Price	0	1
0	2600	5500000	1.0	0.0
1	3000	5650000	1.0	0.0
2	-	-	1.0	0.0
3	-	-	1.0	0.0
4	-	-	0.0	0.0
5	-	-	0.0	0.0
6	-	-	0.0	0.0
7	-	-	0.0	0.0
8	-	-	0.0	1.0
9	-	-	0.0	1.0
10	-	-	0.0	1.0
11	-	-	0.0	1.0

Alphabetical Order

⇒ Label Encoder → Ordinal Variable [sklearn]

Used For binary category Variable

dfnew = df.copy() → Creating "Copy" From original Data.

dfnew

Out

	town	Area	Price
0	Chennai	2600	5500000
1	chennai	3000	5650000
2	Chennai	3200	61000000
:	:	:	:
11	Hyderabad	3600	6950000

From sklearn.preprocessing Import LabelEncoder

Le = LabelEncoder()

dfnew.town = Le.fit_transform (dfnew.town)

dfnew

Out :

	town	Area	Price
0	1		
1	1		
2	1		
3	1		
4	0		
5	0		
6	0		
7	0		
8	2		
9	2		
10	2		
11	2		

Alphabetical Order
↓

Banglore	[B] —	0
Chennai	[C] —	1
Hyderabd	[H] —	2

↓ User Desired values we use this.

⇒ map() → Ordinal Variable [Pandas]

map function.

```
# df_m = df.copy()
# df_m ["town"] = df_m ["town"]. map ({ "chennai :0,
                    "Bangalore ":2 , "hyderabad": 1})

# df_m
```

Out :-

	town	Area	Price
0	0		
1	0		
2	0		
3	0		
4	2		
5	2		
6	2		
7	2		
8	1		
9	1		
10	1		
11	1		

Taken by our choices Here

We can Replace with any Value

chennai	= 0	, 20
Banglore	= 2	, 15
hyderabad	= 1	, 25

which Ever order we write it Takes in The order

⇒ Ordinal Encoder → Ordinal Variable (Ascending order)

From Sklearn. Preprocessing import OrdinalEncoder

```
# df_new1 = df.copy()
# oe = Ordinal Encoder (categories = [[ "Banglore", "Hyderabad", "chennai"]])
                                          0            1             2
# df_new1. town = oe. fit_transform (df_new1 [["town"]])

# df_new1
```

Out :-

	0	1	2	3	4	5	6	7	8	9	10	11
Town	2.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0

6/4/22
3:00AM