Feature Encoding

Feature encoding is the process of transforming categorical features into numeric features. This is necessary because machine learning algorithms can only handle numeric features. There are many different ways to encode categorical features, and each method has its own advantages and disadvantages. In this notebook, we will explore some of the most popular methods for encoding categorical features, such as:

1: Label encoding

2: Ordinal encoding

3: One-hot encoding

4: Binary encoding

In [1]:

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# data load
df = sns.load_dataset('tips')
df.head()
```

Out[1]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [2]:

```
df['time'].value_counts()
```

Out[2]:

```
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

In [3]:

```
# let's encode the time in labelencoder with sklearn

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, OrdinalEncoder
le = LabelEncoder()
df['encoded_time'] = le.fit_transform(df['time'])
df.head()
```

Out[3]:

	total_bill	tip	sex	smoker	day	time	size	encoded_time
0	16.99	1.01	Female	No	Sun	Dinner	2	0
1	10.34	1.66	Male	No	Sun	Dinner	3	0
2	21.01	3.50	Male	No	Sun	Dinner	3	0
3	23.68	3.31	Male	No	Sun	Dinner	2	0
4	24.59	3.61	Female	No	Sun	Dinner	4	0

In [4]:

```
df['encoded_time'].value_counts()
```

Out[4]:

```
encoded_time
0    176
1     68
Name: count, dtype: int64
```

In [5]:

```python
df['day'].value_counts()
```

Out[5]:

```
day
Sat     87
Sun     76
Thur    62
Fri     19
Name: count, dtype: int64
```

In [6]:

```python
# ordinal encoding the day column using specific order
oe = OrdinalEncoder(categories=[['Thur', 'Fri', 'Sat', 'Sun']])
df['encoded_day'] = oe.fit_transform(df[['day']])
df.head()
```

Out[6]:

	total_bill	tip	sex	smoker	day	time	size	encoded_time	encoded_day
0	16.99	1.01	Female	No	Sun	Dinner	2	0	3.0
1	10.34	1.66	Male	No	Sun	Dinner	3	0	3.0
2	21.01	3.50	Male	No	Sun	Dinner	3	0	3.0
3	23.68	3.31	Male	No	Sun	Dinner	2	0	3.0
4	24.59	3.61	Female	No	Sun	Dinner	4	0	3.0

In [7]:

```python
df['encoded_day'].value_counts()
```

Out[7]:

```
encoded_day
2.0    87
3.0    76
0.0    62
1.0    19
Name: count, dtype: int64
```

In [8]:

```python
# one hot encoding on day column
ohe = OneHotEncoder()
```

```
ohe.fit_transform(df[['sex']]).toarray()
```

```
Out[8]: array([[1., 0.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [1., 0.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
               [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.]])
```

In [9]:

```python
# example of one hot encoding
titanic = sns.load_dataset('titanic')
titanic.head()
```

Out[9]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alon
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	Fals
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	Tru
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	Tru

In [10]:

```
# example of one hot encoding
titanic = sns.load_dataset('titanic')

onehot_encoder = OneHotEncoder(sparse=False)
embarked_onehot = onehot_encoder.fit_transform(titanic[['embarked']])
embarked_onehot_df = pd.DataFrame(embarked_onehot, columns=onehot_encoder.get_feature_names_out(['embarked']))
titanic = pd.concat([titanic.reset_index(drop=True), embarked_onehot_df.reset_index(drop=True)], axis=1)
titanic.head()
```

Out[10]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alon
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	Fals
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	Tru
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	Tru

In [13]:

```
# !pip install category_encoders
```

In [12]:

```
df = sns.load_dataset('tips')
df.head()
```

Out[12]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [14]:

```
from category_encoders import BinaryEncoder

binary_encoder = BinaryEncoder()
df_binary = binary_encoder.fit_transform(df['day'])
```

In [15]:

```
# use pandas for feature encoding

df = sns.load_dataset('tips')
df.head()
```

Out[15]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [16]:

```
# use pandas get dummies
get_dummies = pd.get_dummies(df, columns=['day'])
get_dummies.head()
```

Out[16]:

	total_bill	tip	sex	smoker	time	size	day_Thur	day_Fri	day_Sat	day_Sun
0	16.99	1.01	Female	No	Dinner	2	False	False	False	True
1	10.34	1.66	Male	No	Dinner	3	False	False	False	True
2	21.01	3.50	Male	No	Dinner	3	False	False	False	True
3	23.68	3.31	Male	No	Dinner	2	False	False	False	True
4	24.59	3.61	Female	No	Dinner	4	False	False	False	True

In [ ]: