Q1 As for differences in OrdinalEncoder and LabelEncoder implementation, the accepted answer mentions the shape of the data:

OrdinalEncoder is for 2D data with the shape (n_samples, n_features) LabelEncoder is for 1D data with the shape (n_samples,) OrdinalEncoder is for the "features" (often a 2D array), whereas LabelEncoder is for the "target variable" (often a 1D array). Another difference between the encoders is the name of their learned parameter; LabelEncoder learns classes_ OrdinalEncoder learns categories_ LabelEncoder.fit(...) accepts a 1D array; LabelEncoder.classes_ is 1D OrdinalEncoder.fit(...) accepts a 2D array; OrdinalEncoder.categories_ is 2D.

Q2 Target-guided ordinal encoding involves replacing the categories in the categorical variable with ordinal numbers that are based on the relationship between the category and the target variable. Target encoding replaces a categorical value by a blend of the probability (or expected value) of the target given the category with the target probability (or expected value) over the entire training set. Example of how target-guided ordinal encoding can be applied: Let’s say we have a dataset that contains information about employees at a company. One of the variables in the dataset is “job level”, which is a categorical variable with four categories: junior, intermediate, senior, and executive. The target variable in this case is the employee’s salary. To encode the “job level” variable using target-guided ordinal encoding, we would first calculate the mean salary for each job level category. Let’s say the mean salaries are as follows: Junior: $40,000 Intermediate: $60,000 Senior: $80,000 Executive: $120,000 Next, we would sort the job levels based on their mean salaries, from lowest to highest. Then, we would assign ordinal numbers to each job level based on their rank: Junior: 1 Intermediate: 2 Senior: 3 Executive: 4 Now, we have encoded the “job level” variable using target-guided ordinal encoding, and we can use these ordinal numbers as input features in a machine learning model to predict employee salaries. This encoding technique takes into account the relationship between the job level categories and the target variable, which can help improve the accuracy of the model.

Q3 Covariance is a measure of the relationship between two random variables and to what extent, they change together. Or we can say, in other words, it defines the changes between the two variables, such that change in one variable is equal to change in another variable.If cov(X, Y) is greater than zero, then we can say that the covariance for any two variables is positive and both the variables move in the same direction. If cov(X, Y) is less than zero, then we can say that the covariance for any two variables is negative and both the variables move in the opposite direction.If cov(X, Y) is zero, then we can say that there is no relation between two variables. Correlation,ρ(X,Y) = Cov(X,Y)/σX σy EX:-

Cov(X, Y) = Σ(Xi-µ)(Yj-v) / n. 6,911.45 + 25.95 + 1,180.85 + 28.35 + 906.95 + 9,837.45 = 18,891. Cov(X, Y) = 18,891 / 6

In [33]:
```
#Q4
# Import pandas library
import pandas as pd

# initialize list elements
#data = ['Red','Green','Yellow']
#data2=['small','mediaum','large']
#data3=['wood','metal','plastic']

# Create the pandas DataFrame with column name is provided explicitly
df = pd.DataFrame({'color':['R','Y','B'],
                  'Size':['S','M','L'],
                  'Material':['w','m','p']})


# print dataframe.
df
```

Out[33]:

	color	Size	Material
0	R	S	w
1	Y	M	m
2	B	L	p

In [34]:
```
!pip install category_encoders
import category_encoders as ce
```

```
Requirement already satisfied: category_encoders in /opt/conda/lib/python3.10/site
-packages (2.6.2)
Requirement already satisfied: patsy>=0.5.1 in /opt/conda/lib/python3.10/site-pack
ages (from category_encoders) (0.5.3)
Requirement already satisfied: pandas>=1.0.5 in /opt/conda/lib/python3.10/site-pac
kages (from category_encoders) (1.5.2)
Requirement already satisfied: statsmodels>=0.9.0 in /opt/conda/lib/python3.10/sit
e-packages (from category_encoders) (0.13.5)
Requirement already satisfied: numpy>=1.14.0 in /opt/conda/lib/python3.10/site-pac
kages (from category_encoders) (1.23.5)
Requirement already satisfied: scikit-learn>=0.20.0 in /opt/conda/lib/python3.10/s
ite-packages (from category_encoders) (1.2.0)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.10/site-pack
ages (from category_encoders) (1.9.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/conda/lib/python3.1
0/site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-pack
ages (from pandas>=1.0.5->category_encoders) (2022.6)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages (fro
m patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /opt/conda/lib/python3.10/site-pac
kages (from scikit-learn>=0.20.0->category_encoders) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.10/s
ite-packages (from scikit-learn>=0.20.0->category_encoders) (3.1.0)
Requirement already satisfied: packaging>=21.3 in /opt/conda/lib/python3.10/site-p
ackages (from statsmodels>=0.9.0->category_encoders) (22.0)
```

```
In [35]: encoder=ce.OneHotEncoder(cols='color',handle_unknown='return_nan',return_df=True,us
         encoder=ce.OneHotEncoder(cols='Size',handle_unknown='return_nan',return_df=True,use
         encoder=ce.OneHotEncoder(cols='Material',handle_unknown='return_nan',return_df=True
         #Original Data
         print(df)
         #Fit and transform Data
         df_encoded = encoder.fit_transform(df)
         print(df_encoded)
```

```
  color Size Material
0     R    S        w
1     Y    M        m
2     B    L        p
  color Size  Material_w  Material_m  Material_p
0     R    S         1.0         0.0         0.0
1     Y    M         0.0         1.0         0.0
2     B    L         0.0         0.0         1.0
```

Q5 Sample covariance matrix is given by \frac{\sum_{1}^{n}\left (x_{i} -\overline{x}\right)^{2} }{n-1} .

n = 5, µx = 22.4, var(X) = 321.2 / (5 – 1) = 80.3

µy = 12.58, var(Y) = 132.148 / 4 = 33.037

µz = 64, var(Z) = 570 / 4 = 142.5

cov(X, Y) = \frac{\sum_{1}^{5}\left (x_{i} -22.4\right)\left (y_{i}-12.58\right) }{5-1} = -11.76

cov(X, Z) = \frac{\sum_{1}^{5}\left (x_{i} -22.4\right)\left (z_{i}-64 \right) }{5-1} = 34.97

cov(Y, Z) = \frac{\sum_{1}^{5}\left (y_{i} -12.58\right)\left (z_{i}-64 \right) }{5-1} = -40.87

\begin{bmatrix} 80.3 & -13.865 &14.25 \\ -13.865 & 33.037 & -39.5250\\ 14.25 & -39.5250 & 142.5 \end{bmatrix}

Q6 In this machine learning project we use One hot encoding to each and every variable It allows the use of categorical variables in models that require numerical input. It can improve model performance by providing more information to the model about the categorical variable. It can help to avoid the problem of ordinality, which can occur when a categorical variable has a natural ordering (e.g. “small”, “medium”, “large”). However, various Machine Learning models do not work with categorical data and to fit this data into the machine learning model it needs to be converted into numerical data. For example, suppose a dataset has a Gender column with categorical elements like Male and Female. These labels have no specific order of preference and also since the data is string labels, machine learning models misinterpreted that there is some sort of hierarchy in them.

In [41]:
```
#Q7
import numpy as np
import pandas as pd
import seaborn as sns
```

In [43]:
```
df=sns.load_dataset("iris")
```

In [46]:
```
df.cov()
```

```
/tmp/ipykernel_103/1545644723.py:1: FutureWarning: The default value of numeric_on
ly in DataFrame.cov is deprecated. In a future version, it will default to False.
Select only valid columns or specify the value of numeric_only to silence this war
ning.
  df.cov()
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	0.685694	-0.042434	1.274315	0.516271
sepal_width	-0.042434	0.189979	-0.329656	-0.121639
petal_length	1.274315	-0.329656	3.116278	1.295609
petal_width	0.516271	-0.121639	1.295609	0.581006

In [ ]: