In [1]:

```
# full code in one cell


import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

df = pd.read_csv('iris.csv')

print("Missing values before filling:\n", df.isnull().sum())

df['sepal_length'].fillna(df['sepal_length'].mean(), inplace=True)
df['sepal_width'].fillna(df['sepal_width'].mean(), inplace=True)
df['petal_length'].fillna(df['petal_length'].mean(), inplace=True)
df['petal_width'].fillna(df['petal_width'].mean(), inplace=True)
df['species'].fillna(df['species'].mode()[0], inplace=True)

print("\nMissing values after filling:\n", df.isnull().sum())

print("\nData types before conversion:\n", df.dtypes)

df['species'] = df['species'].astype('category')

print("\nData types after conversion:\n", df.dtypes)

le = LabelEncoder()
df['species_encoded'] = le.fit_transform(df['species'])

scaler = MinMaxScaler()
numerical_cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

print("\nFinal DataFrame:\n", df.head())
```

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler,LabelEncoder
```

In [2]:

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

In [3]:

```
df.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [4]:

```
df.isnull().sum()
 # preprocessing
```

Out[4]:

```
sepal_length    2
sepal_width     1
petal_length    2
petal_width     1
species         3
dtype: int64
```

In [6]:

```
df['sepal_length'].fillna(df['sepal_length'].mean(), inplace=True)
df['sepal_width'].fillna(df['sepal_width'].mean(), inplace=True)
df['petal_length'].fillna(df['petal_length'].mean(), inplace=True)
df['petal_width'].fillna(df['petal_width'].mean(), inplace=True)
```

```
C:\Users\Chatura Karankal\AppData\Local\Temp\ipykernel_7324\800527030.py:1: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['sepal_length'].fillna(df['sepal_length'].mean(), inplace=True)
C:\Users\Chatura Karankal\AppData\Local\Temp\ipykernel_7324\800527030.py:2: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['sepal_width'].fillna(df['sepal_width'].mean(), inplace=True)
C:\Users\Chatura Karankal\AppData\Local\Temp\ipykernel_7324\800527030.py:3: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['petal_length'].fillna(df['petal_length'].mean(), inplace=True)
C:\Users\Chatura Karankal\AppData\Local\Temp\ipykernel_7324\800527030.py:4: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['petal_width'].fillna(df['petal_width'].mean(), inplace=True)
```

In [7]:

```
df['species'].fillna(df['species'].mode()[0], inplace=True)
```

```
C:\Users\Chatura Karankal\AppData\Local\Temp\ipykernel_7324\2314026490.py:1: Futu
reWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['species'].fillna(df['species'].mode()[0], inplace=True)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [9]:
```
df.describe()
```

Out[9]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.829054	3.057718	3.771622	1.205369
std	0.818843	0.431196	1.751089	0.758733
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.300000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [10]:
```
df.columns.tolist()
```

Out[10]:
```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

In [11]:
```
df.shape
```

Out[11]:
```
(150, 5)
```

In [12]:
```
df.dtypes
```

Out[12]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [13]:

```
df['species']=df['species'].astype('category')       #data formatting
```

In [14]:

```
df.dtypes
```

Out[14]:

```
sepal_length     float64
sepal_width      float64
petal_length     float64
petal_width      float64
species         category
dtype: object
```

In [15]:

```
scaler=MinMaxScaler()#   normalization
cols=['sepal_length','sepal_width','petal_length','petal_width']
df[cols]=scaler.fit_transform(df[cols])
df.head()
```

Out[15]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	0.222222	0.625000	0.067797	0.041667	setosa
1	0.166667	0.416667	0.067797	0.041667	setosa
2	0.111111	0.500000	0.050847	0.041667	setosa
3	0.083333	0.458333	0.084746	0.041667	setosa
4	0.194444	0.666667	0.067797	0.041667	setosa

In [16]:

```
label_encoder=LabelEncoder()
```

In [18]:

```
df['species_encoded']=label_encoder.fit_transform(df['species'])
```

In [19]:

```
df.head()
```

Out[19]:

	sepal_length	sepal_width	petal_length	petal_width	species	species_encoded
0	0.222222	0.625000	0.067797	0.041667	setosa	0
1	0.166667	0.416667	0.067797	0.041667	setosa	0
2	0.111111	0.500000	0.050847	0.041667	setosa	0
3	0.083333	0.458333	0.084746	0.041667	setosa	0
4	0.194444	0.666667	0.067797	0.041667	setosa	0

In [ ]: