Hamna Jamila

AKSA- AI Internship

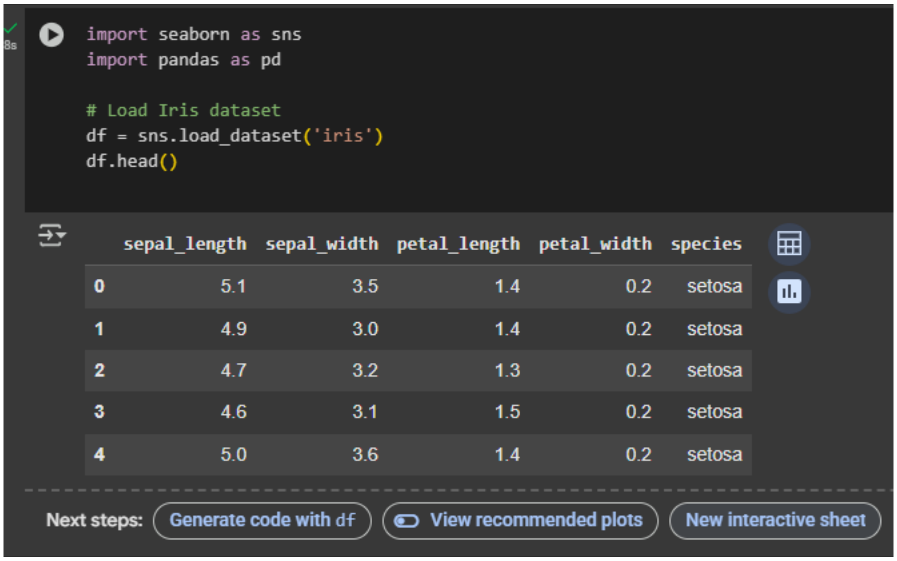
Task: Feature Engineering

What is Feature Engineering?

Feature Engineering is the process of transforming raw data into meaningful inputs for machine learning models. It includes handling missing values, encoding, scaling, creating new features, and more.

I’ll use the **iris dataset** as the example for the implementation of feature engineering task.

Importing the dataset



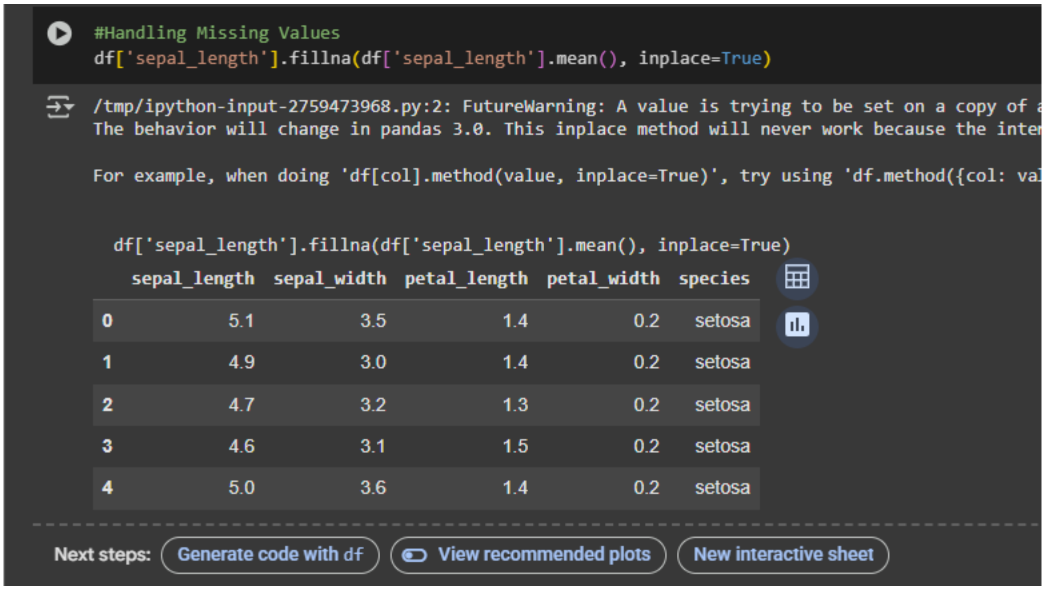
```
import seaborn as sns
import pandas as pd

# Load Iris dataset
df = sns.load_dataset('iris')
df.head()
```

	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

Feature Engineering Techniques

1. Handling Missing Values

Iris actually has no missing values though, but if there were, following is the example code how will we be handling them.



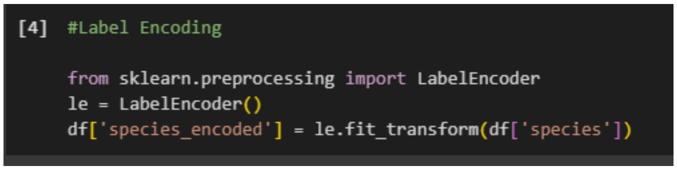
```
#Handling Missing Values
df['sepal_length'].fillna(df['sepal_length'].mean(), inplace=True)
```

```
/tmp/ipython-input-2759473968.py:2: FutureWarning: A value is trying to be set on a copy of a
The behavior will change in pandas 3.0. This inplace method will never work because the inter

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val


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

	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

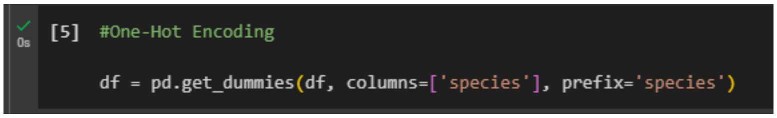
2. Encoding Categorical Variables

Label Encoding



```
[4] #Label Encoding

    from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    df['species_encoded'] = le.fit_transform(df['species'])
```

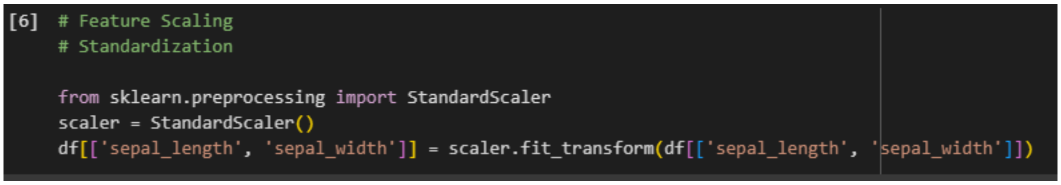
One-Hot Encoding



```
[5] #One-Hot Encoding

    df = pd.get_dummies(df, columns=['species'], prefix='species')
```

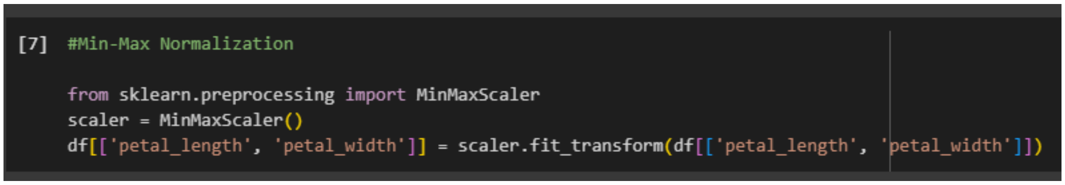
3. Feature Scaling

Standardization



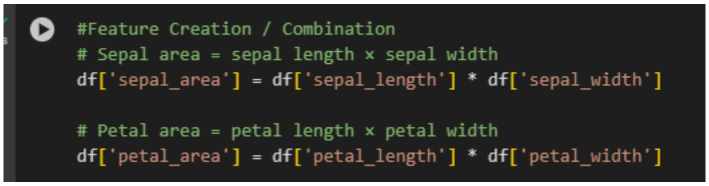
```
[6] # Feature Scaling
    # Standardization

    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    df[['sepal_length', 'sepal_width']] = scaler.fit_transform(df[['sepal_length', 'sepal_width']])
```

Min-Max Normalization



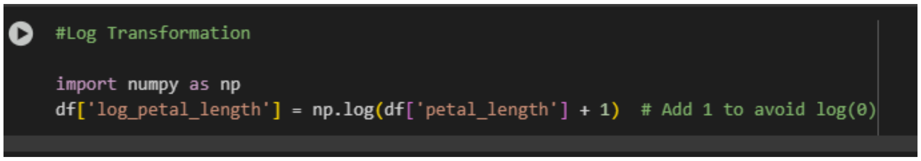
```
#Min-Max Normalization

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['petal_length', 'petal_width']] = scaler.fit_transform(df[['petal_length', 'petal_width']])
```

4. Feature Creation / Combination



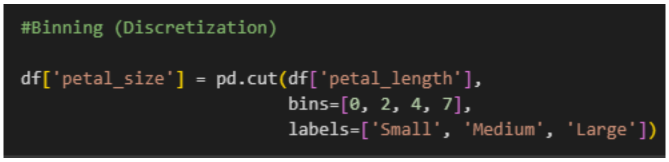
```
#Feature Creation / Combination
# Sepal area = sepal length × sepal width
df['sepal_area'] = df['sepal_length'] * df['sepal_width']

# Petal area = petal length × petal width
df['petal_area'] = df['petal_length'] * df['petal_width']
```

5. Log Transformation



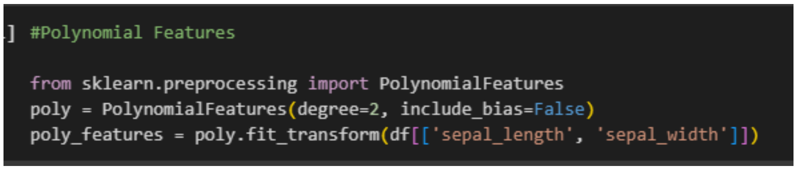
```
#Log Transformation

import numpy as np
df['log_petal_length'] = np.log(df['petal_length'] + 1)  # Add 1 to avoid log(0)
```

6. Binning (Discretization)



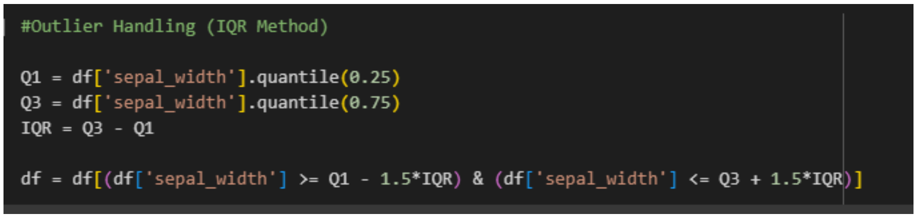
```
#Binning (Discretization)

df['petal_size'] = pd.cut(df['petal_length'],
                          bins=[0, 2, 4, 7],
                          labels=['Small', 'Medium', 'Large'])
```

7. Polynomial Features



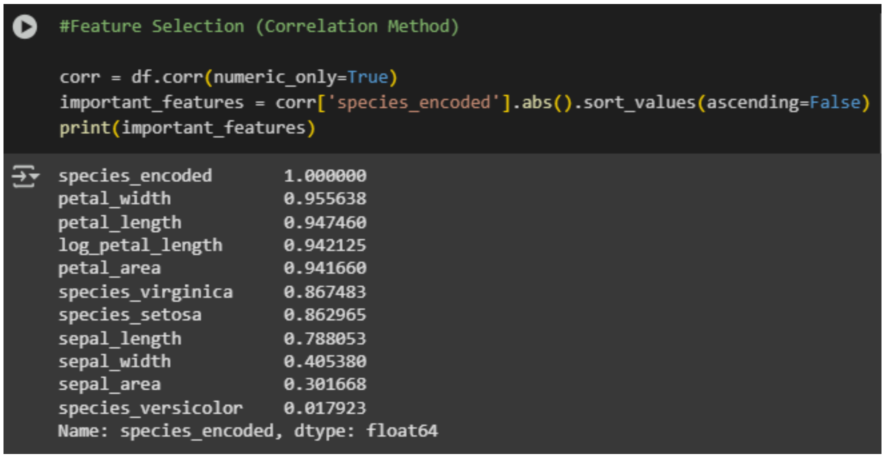
```
#Polynomial Features

from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(df[['sepal_length', 'sepal_width']])
```

8. Outlier Handling (IQR Method)



```
#Outlier Handling (IQR Method)

Q1 = df['sepal_width'].quantile(0.25)
Q3 = df['sepal_width'].quantile(0.75)
IQR = Q3 - Q1

df = df[(df['sepal_width'] >= Q1 - 1.5*IQR) & (df['sepal_width'] <= Q3 + 1.5*IQR)]
```

9. Feature Selection (Correlation Method)



```
#Feature Selection (Correlation Method)

corr = df.corr(numeric_only=True)
important_features = corr['species_encoded'].abs().sort_values(ascending=False)
print(important_features)
```

```
species_encoded       1.000000
petal_width           0.955638
petal_length          0.947460
log_petal_length      0.942125
petal_area            0.941660
species_virginica     0.867483
species_setosa        0.862965
sepal_length          0.788053
sepal_width           0.405380
sepal_area            0.301668
species_versicolor    0.017923
Name: species_encoded, dtype: float64
```

Summary:

Technique	**Purpose**	**Example (Iris Dataset)**
Imputation	Fill missing data	fillna(df.mean())
Encoding	Convert categories to numbers	LabelEncoder, get_dummies()
Scaling	Normalize feature range	StandardScaler, MinMaxScaler
Feature Creation	New features from existing ones	sepal_area = length × width
Log Transformation	Reduce skewness	np.log(petal_length + 1)
Binning	Convert continuous to discrete	pd.cut() for petal_length
Polynomial Features	Add interaction/power features	PolynomialFeatures()
Outlier Handling	Remove extreme values	IQR method on sepal_width
Feature Selection	Choose important features	Based on correlation