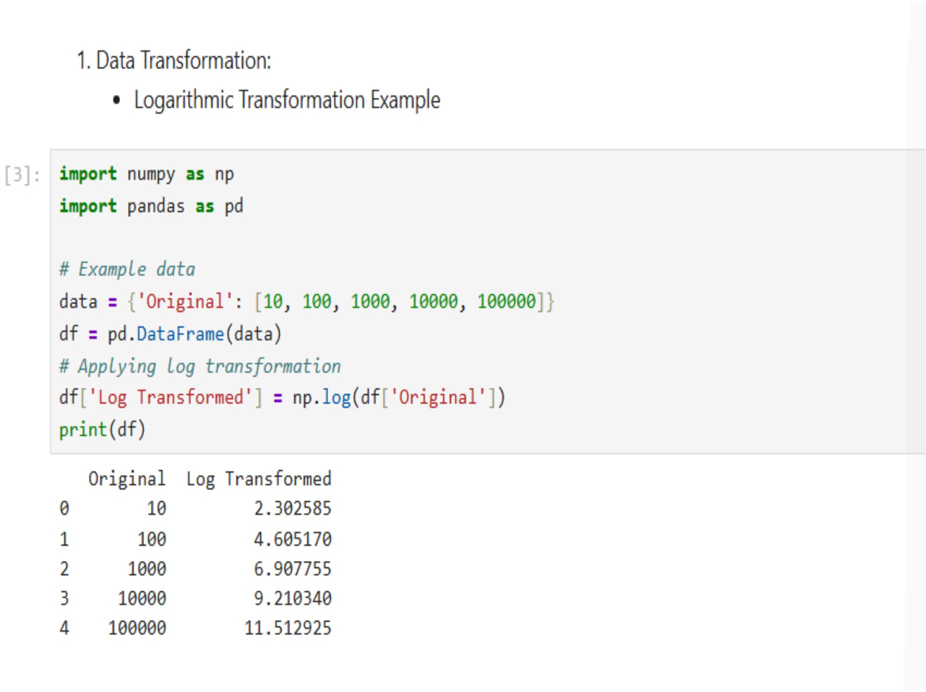
Villanueva, Ashley Nicole N.

BSIT – 3B

IT 314 - Data Analytics

DATA TRANSFORMATION AND NORMALIZATION

1. Data Transformation – Logarithmic Transformation



1. Data Transformation:

- Logarithmic Transformation Example

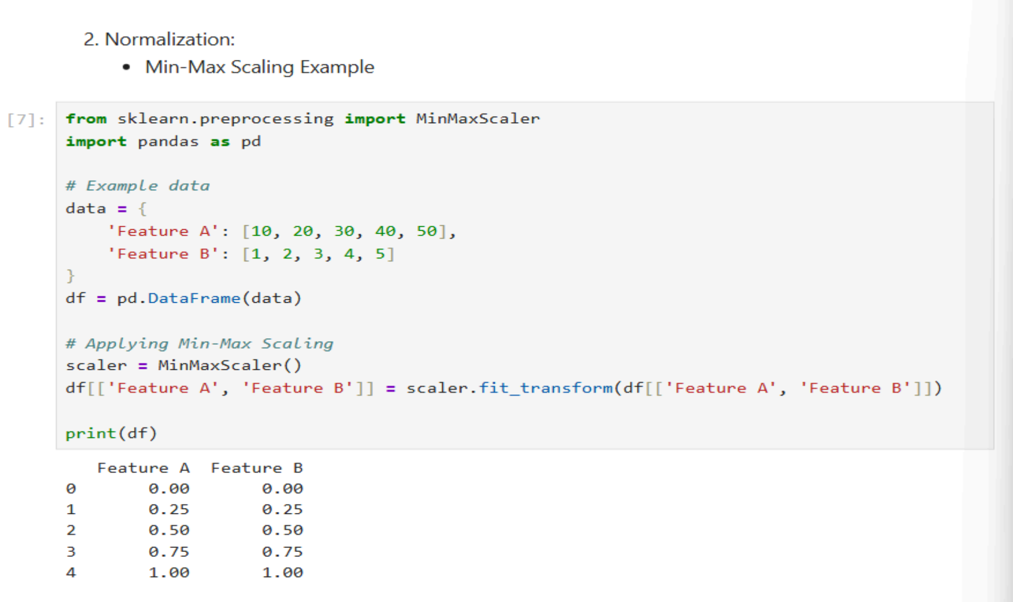
```
[3]: import numpy as np
     import pandas as pd

     # Example data
     data = {'Original': [10, 100, 1000, 10000, 100000]}
     df = pd.DataFrame(data)
     # Applying log transformation
     df['Log Transformed'] = np.log(df['Original'])
     print(df)
```

```
   Original  Log Transformed
0        10         2.302585
1       100         4.605170
2      1000         6.907755
3     10000         9.210340
4    100000        11.512925
```

EXPLANATION:

- Data with high variance can reduce the impact which makes it easier to work with certain models.
- np.log() applies the natural logarithm (base *e*) to each value.
- Large values are compressed, and small values are expanded.

2. Normalization – Min-Max Scaling



2. Normalization:

- Min-Max Scaling Example

[7]:

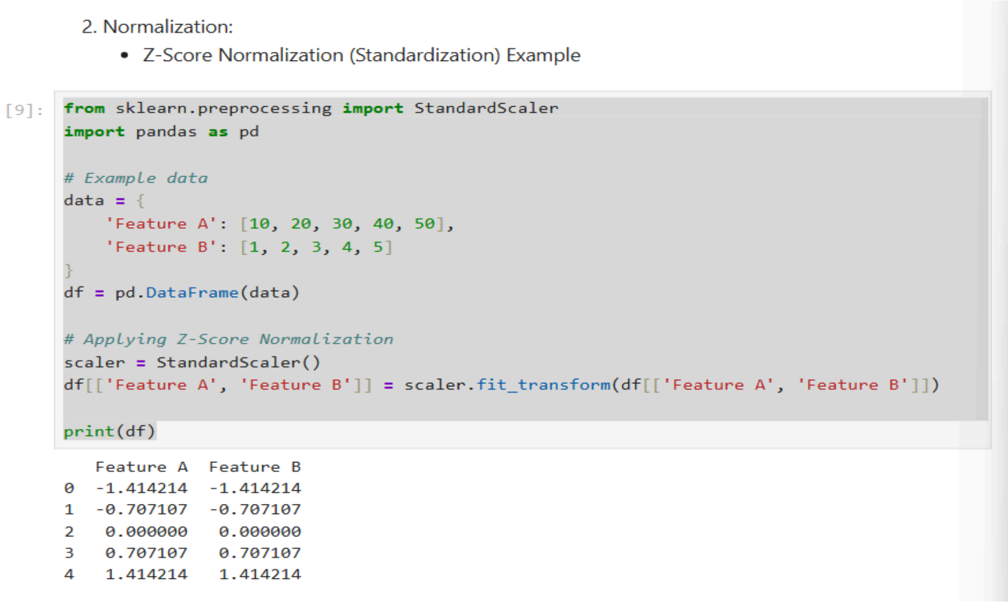
```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd

# Example data
data = {
    'Feature A': [10, 20, 30, 40, 50],
    'Feature B': [1, 2, 3, 4, 5]
}
df = pd.DataFrame(data)

# Applying Min-Max Scaling
scaler = MinMaxScaler()
df[['Feature A', 'Feature B']] = scaler.fit_transform(df[['Feature A', 'Feature B']])

print(df)
```

```
   Feature A  Feature B
0       0.00       0.00
1       0.25       0.25
2       0.50       0.50
3       0.75       0.75
4       1.00       1.00
```

EXPLANATION:

- In the example, both Feature A and Feature B are rescaled to fall within the range [0, 1].
- Min-Max scaling rescales features to **a** range between 0 and 1.

3. Normalization – Z-Score Normalization



2. Normalization:

- Z-Score Normalization (Standardization) Example

[9]:

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Example data
data = {
    'Feature A': [10, 20, 30, 40, 50],
    'Feature B': [1, 2, 3, 4, 5]
}
df = pd.DataFrame(data)

# Applying Z-Score Normalization
scaler = StandardScaler()
df[['Feature A', 'Feature B']] = scaler.fit_transform(df[['Feature A', 'Feature B']])

print(df)
```

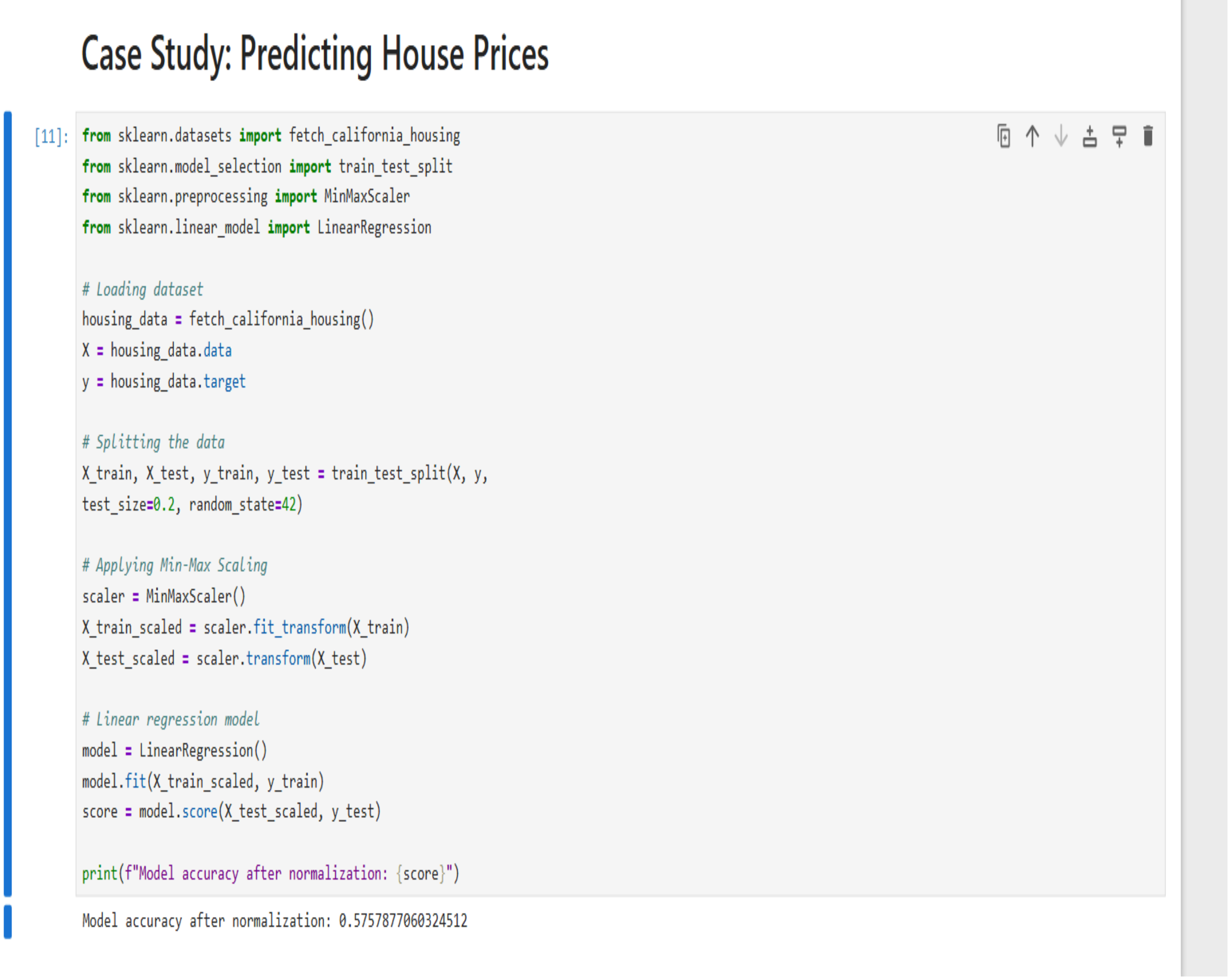
```
   Feature A  Feature B
0  -1.414214  -1.414214
1  -0.707107  -0.707107
2   0.000000   0.000000
3   0.707107   0.707107
4   1.414214   1.414214
```

EXPLANATION:

- StandardScaler transforms data to have mean = 0 and standard deviation = 1.
- After Z-Score normalization, the features will have a mean of 0 and a

standard deviation of 1.

4. **Case Study: Predicting House Prices**



Case Study: Predicting House Prices

[11]:

```python
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression

# Loading dataset
housing_data = fetch_california_housing()
X = housing_data.data
y = housing_data.target

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Applying Min-Max Scaling
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Linear regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)
score = model.score(X_test_scaled, y_test)

print(f"Model accuracy after normalization: {score}")
```

```
Model accuracy after normalization: 0.5757877060324512
```

RESULT: Model accuracy after normalization: 0.5757877060324512

EXPLANATION:

- In this case, normalizing the features using Min-Max scaling improves the model’s performance by ensuring that no feature dominates the learning process due to its scale.
- The dataset has features with different ranges (e.g., population, median income).
- Min-Max scaling ensures all features are between 0 and 1.