การทำ Data Preprocessing

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

# ขั้นตอนที่ 1: โหลดข้อมูลจากไฟล์ CSV
file_path = 'C:/Users/Windows 11/Desktop/dataML/prep-data-lab03.csv'
df = pd.read_csv(file_path)
# อ่านไฟล์ CSV จากตำแหน่งที่กำหนดและเก็บข้อมูลใน DataFrame ของ pandas ชื่อ 'df'

# ขั้นตอนที่ 2: จัดการกับค่า 0 ในคอลัมน์ precipitation เนื่องจาก มีค่า 0 จำนวนมากใน column โดยแทนที่ด้วยค่าเฉลี่ยของคอลัมน์ที่ไม่เป็น 0
# คำนวณค่าเฉลี่ยของ precipitation ที่ไม่เป็น 0
precipitation_mean = df['precipitation'][df['precipitation'] != 0].mean()
# แทนที่ค่าที่เป็น 0 ในคอลัมน์ precipitation ด้วยค่าเฉลี่ยที่คำนวณได้
df['precipitation'] = df['precipitation'].replace(0, precipitation_mean)
```

```python
# ขั้นตอนที่ 3: ทำการ Min-Max Scaling กับฟีเจอร์ (ไม่รวมตัวแปรที่ใช้ทำนาย(Y) 'bicycle_counts')
# ไม่ใช้ Standard เนื่องจากค่ากระจายแบบไม่ปกติทำ standard แล้วค่าจะติดลบ
scaler = MinMaxScaler()
df[['temperature', 'precipitation']] = scaler.fit_transform(df[['temperature', 'precipitation']])
# ใช้ MinMaxScaler เพื่อปรับค่าในคอลัมน์ temperature และ precipitation ให้อยู่ในช่วง [0, 1]

# ขั้นตอนที่ 4: บันทึกข้อมูลที่ผ่านการ preprocess แล้วลงในไฟล์ CSV ใหม่
output_file_path = 'C:/Users/Windows 11/Desktop/dataML/preprocessed_prep-data-lab03.csv'
df.to_csv(output_file_path, index=False)
# บันทึกข้อมูลที่ผ่านการ preprocess แล้วลงในไฟล์ CSV ใหม่ในตำแหน่งที่กำหนด

# ตรวจสอบไฟล์ที่บันทึกโดยการแสดงแถวแรกๆ ของ DataFrame
print(df.head())
```

```
C:\PyLab\pythonProject\.venv\Scripts\python.exe C:\PyLab\pythonProject2\002.py
   day  temperature  precipitation  bicycle_counts
0    1          0.7       0.437964             150
1    2          0.4       0.317073              87
2    3          0.2       0.437964             108
3    4          0.6       0.437964             162
4    5          0.3       0.707317              65

Process finished with exit code 0
```

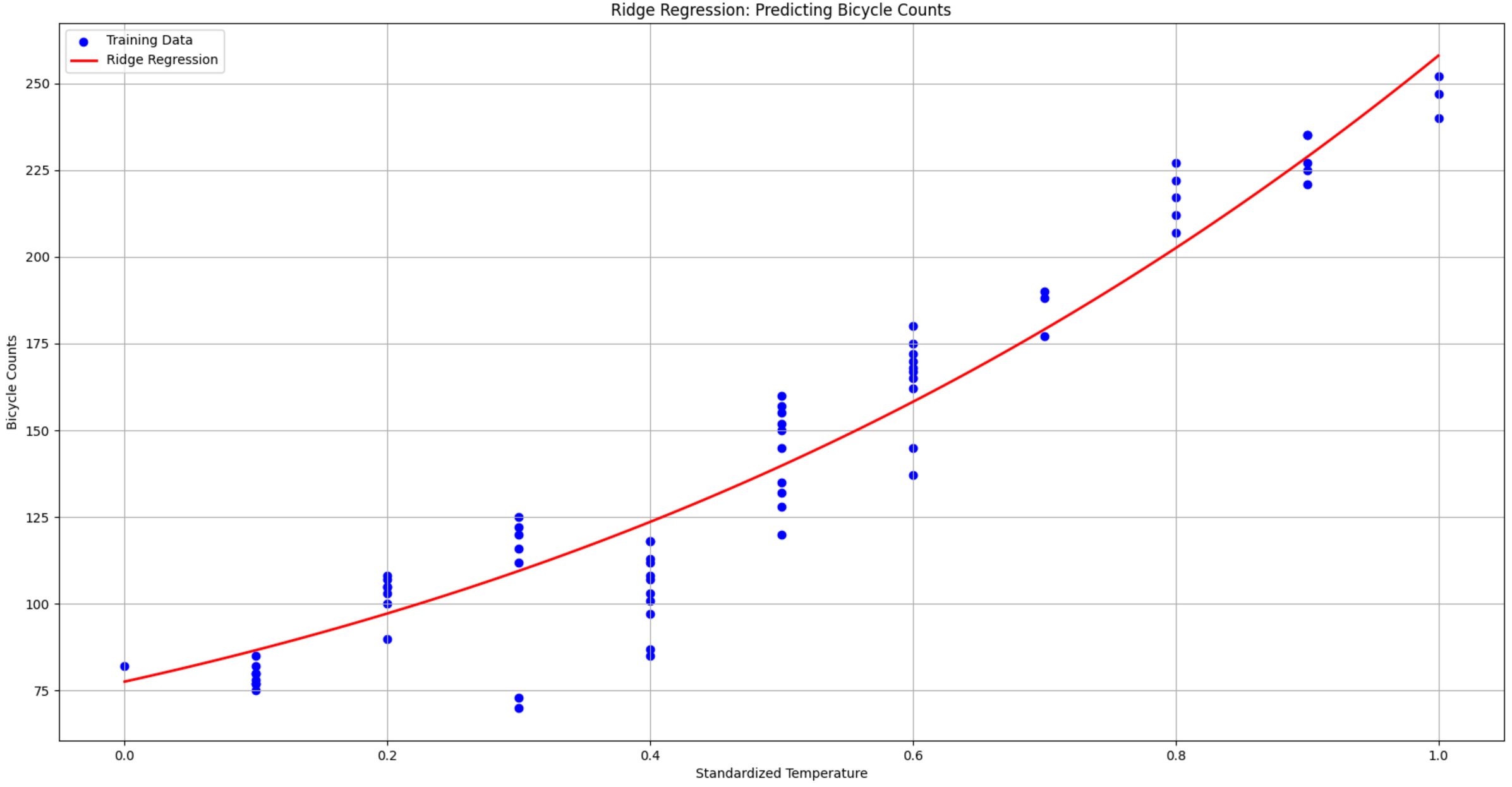
การทำ Model

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# ขั้นตอนที่ 1: โหลดข้อมูลที่ผ่านการ preprocessing แล้วจากไฟล์ CSV
file_path = 'C:/Users/Windows 11/Desktop/dataML/preprocessed_prep-data-lab03.csv'
df = pd.read_csv(file_path)
# อ่านข้อมูลจากไฟล์ CSV ที่กำหนดและเก็บใน DataFrame ชื่อ 'df'


# เลือก column ที่ใช้ train 'temperature', 'precipitation'
# เลือก column ที่ทำนาย  'bicycle_counts'
X = df[['temperature', 'precipitation']].values  # แปลงข้อมูลเป็น numpy array
y = df['bicycle_counts'].values  #ทำนาย
```

```python
# ขั้นตอนที่ 2: แบ่งข้อมูบเป็น (training), (testing), (validation) โดยใช้แบบ train 70% test 20% valid 10%
X_temp, X_test, y_temp, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split( *arrays: X_temp, y_temp, test_size=0.125, random_state=42)



# ขั้นตอนที่ 3: สร้างฟีเจอร์โพลีโนเมียล (Polynomial Features) เนื่องจาก data มีการกระจายเป็นเส้นโค้ง
poly = PolynomialFeatures(degree=3)  # เลือกดีกรีเป็น 2 หรือ 3      (ไม่ควรเกิน 3 เนื่องจากจะใช้ทรัพยากรเครื่องเยอะ)
X_train_poly = poly.fit_transform(X_train)  # สร้างฟีเจอร์train
X_valid_poly = poly.transform(X_valid)  # สร้างฟีเจอร์valid
X_test_poly = poly.transform(X_test)  # สร้างฟีเจอร์test
# ขั้นตอนที่ 4: สร้างโมเดล Ridge Regression
model_ridge = Ridge(alpha=0.5)  # ปรับค่า alpha เพื่อควบคมค่าปรับ (penalty) ให้ได้ผลทำนายที่แม่นยำที่สด
# Ridge Regression ช่วยลดปัญหา overfitting โดยการเพิ่มค่าปรับในสมการ Polynomial Regression ทำให้โมเดลมีความเสถียรมากขึ้น
model_ridge.fit(X_train_poly, y_train)  # การ train model

# ขั้นตอนที่ 5: ทำนายค่าจำนวนจักรยาน (bicycle_counts) โดยใช้โมเดล Ridge Regression
y_train_pred = model_ridge.predict(X_train_poly)
y_valid_pred = model_ridge.predict(X_valid_poly)
y_test_pred = model_ridge.predict(X_test_poly)

# ขั้นตอนที่ 6: การวาดกราฟ (กว้าง,ยาว)
plt.figure(figsize=(20, 10))
```

```python
# พล็อตกราฟแบบกระจายของข้อมูลTrain
plt.scatter(X_train[:, 0], y_train, color='blue', label='Training Data')

# สร้างจุดตามช่วงของ temperature สำหรับกราฟที่ราบรื่นขึ้น
temperature_range = np.linspace(X_train[:, 0].min(), X_train[:, 0].max(), num: 100).reshape(-1, 1)
# สร้าง DataFrame สำหรับช่วง temperature และค่าเฉลี่ยของ precipitation
mean_precipitation = X_train[:, 1].mean()
X_range = np.hstack((temperature_range, np.full_like(temperature_range, mean_precipitation)))
X_range_poly = poly.transform(X_range)
y_range_pred = model_ridge.predict(X_range_poly)

# พล็อตกราฟการทำนายโดยโมเดล Ridge Regression
plt.plot( *args: temperature_range, y_range_pred, color='red', linewidth=2, label='Ridge Regression')

plt.title('Ridge Regression: Predicting Bicycle Counts')
plt.xlabel('Standardized Temperature')
plt.ylabel('Bicycle Counts')
plt.legend()
plt.grid(True)
plt.show()
```

```python
# ขั้นตอนที่ 7: ประเมินผลการทำงานของโมเดล
print("Training Set - Mean Squared Error:", mean_squared_error(y_train, y_train_pred))
print("Training Set - R^2 Score:", r2_score(y_train, y_train_pred))
print("\nValidation Set - Mean Squared Error:", mean_squared_error(y_valid, y_valid_pred))
print("Validation Set - R^2 Score:", r2_score(y_valid, y_valid_pred))
print("\nTesting Set - Mean Squared Error:", mean_squared_error(y_test, y_test_pred))
print("Testing Set - R^2 Score:", r2_score(y_test, y_test_pred))


#train set คือ เราเทรนโมเดลมาได้แม่นเท่าไหร่
#valid set คือ เราป่าโมเดลที่เทรนมาใช้ตรวจสอบความแม่นยำ
#test set คือ เราป่าโมเดลที่เทรนมาใช้ทดสอบความแม่นยำ
```


Ridge Regression: Predicting Bicycle Counts
Training Data
Ridge Regression
Bicycle Counts
Standardized Temperature
250
225
200
175
150
125
100
75
0.0
0.2
0.4
0.6
0.8
1.0

```
Training Set - Mean Squared Error: 161.62074897771907
Training Set - R^2 Score: 0.9399323195211872

Validation Set - Mean Squared Error: 112.31863090737468
Validation Set - R^2 Score: 0.947169283819279

Testing Set - Mean Squared Error: 258.5195895465106
Testing Set - R^2 Score: 0.902218418593920 2
```