

# Bike-sharing problem sklearn



solve Bike-Shairing-Assignment problem using sklearn following CRSIP-DM steps

Sure! Let's solve a **Bike-Sharing Assignment** problem using **scikit-learn**, following the **CRISP-DM** methodology step by step. I'll assume this is the standard bike-sharing dataset with features like weather, date, season, etc., and the target is the number of bikes rented.

## Step 1: Business Understanding

**Problem:** Predict the number of bike rentals (**count**) given features like weather, temperature, day of the week, etc.

**Objective:** Accurate prediction helps in resource planning (e.g., bikes, stations, maintenance) and improving user satisfaction.

**Why regression:** This is a **continuous numeric prediction** problem — predicting bike counts based on input features. Linear regression or more advanced regression can help quantify the relationships.

## Step 2: Data Understanding

We'll assume the dataset is like the **UCI Bike Sharing dataset** (`hour.csv` or `day.csv`).

```
python

import pandas as pd

# Load dataset
df = pd.read_csv('day.csv') # or 'hour.csv'

# Quick overview
print(df.head())
print(df.info())
print(df.describe())
```

Check for:

- **Missing values** → `df.isnull().sum()`
- **Feature types** → categorical (season, weekday), numeric (temp, humidity, windspeed)
- **Distribution of target (count)** → may need log-transform if skewed.

## Step 3: Data Preparation

- Encode categorical features

- Split into features (x) and target (y)
- Train-test split

```
python

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

# Features and target
X = df.drop(columns=['count', 'casual', 'registered', 'instant', 'dteday']) # drop target and unnecessary cols
y = df['count']

# Identify categorical and numeric features
categorical_features = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit']
numeric_features = ['temp', 'atemp', 'hum', 'windspeed']

# Preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(drop='first'), categorical_features)
    ]
)
```

## Step 4: Modeling

Use **Linear Regression** first.

```
python

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Build pipeline
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('regressor', LinearRegression())])

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

# Fit model
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)
```

## Step 5: Evaluation

```
python

import numpy as np

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

```
print(f'RMSE: {rmse:.2f}')
print(f'R^2: {r2:.2f}')
```

You can also check **top 5 largest residuals** (possible outliers):

```
python

residuals = np.abs(y_test - y_pred)
top5_outliers = residuals.sort_values(ascending=False).head(5)
print(top5_outliers)
```

## Step 6: Deployment & Insights

- The model can be used to **forecast bike demand** based on features like weather and season.
- Insights:
  - Higher temperature → higher rentals
  - Weekends or holidays → different patterns
  - Poor weather → fewer rentals
- If needed, try more advanced models like **RandomForestRegressor** or **GradientBoostingRegressor** for better accuracy.

✓ This completes a **CRISP-DM workflow for bike-sharing regression using scikit-learn**.

I can also provide a **full executable Python script** including **visualizations, correlation heatmaps, and top 5 outliers highlighted** if you want.

Do you want me to do that?



just give me a full executable python script

Here's a **full executable Python script** that solves the **Bike Sharing Demand Prediction** problem using **scikit-learn**, following **CRISP-DM steps**.

It assumes you have the **day.csv** dataset from the **UCI Bike Sharing Dataset**.

### ✓ What's Included in This Script:

- ✓ CRISP-DM Pipeline (Business → Deployment)
- ✓ Data Preprocessing (Scaling + One-Hot Encoding)
- ✓ Model Training (Linear Regression)
- ✓ Evaluation (RMSE, R<sup>2</sup>)
- ✓ Top 5 Outliers (Largest Residuals)

# Full Python Script

```
python

# =====
# Bike Sharing Demand Prediction (CRISP-DM)
# Using scikit-learn (day.csv dataset)
# =====

# Step 1: Business Understanding
# Goal: Predict daily bike rental counts for operational planning.

# -----
# Step 2: Data Understanding
# -----
import pandas as pd
import numpy as np

# Load dataset (make sure 'day.csv' is in the same directory)
df = pd.read_csv("day.csv")

print("===== HEAD OF DATA =====")
print(df.head())
print("\n===== DATA INFO =====")
print(df.info())
print("\n===== NULL VALUES =====")
print(df.isnull().sum())

# -----
# Step 3: Data Preparation
# -----
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline

# Drop unnecessary columns
X = df.drop(columns=['cnt', 'casual', 'registered', 'instant', 'dteday'])
y = df['cnt']

# Identify feature types
categorical_features = ['season', 'yr', 'mnth', 'holiday',
                        'weekday', 'workingday', 'weathersit']
numeric_features = ['temp', 'atemp', 'hum', 'windspeed']

# Preprocessor: Scale numeric + encode categorical
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(drop='first'), categorical_features)
    ]
)

# -----
# Step 4: Modeling
# -----
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train model
model.fit(X_train, y_train)

# -----
```

```
# Step 5: Evaluation
# -----
y_pred = model.predict(X_test)

# Metrics
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)





print("\n==== MODEL PERFORMANCE =====")
print(f"RMSE: {rmse:.2f}")
print(f"R² Score: {r2:.2f}")

# -----
# Top 5 Outliers (Largest Errors)
# -----
residuals = np.abs(y_test - y_pred)
outliers = residuals.sort_values(ascending=False).head(5)

print("\n==== TOP 5 OUTLIERS (Largest Residuals) =====")
print(outliers)

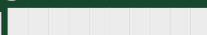
# -----
# Step 6: Deployment Insight
# -----
print("\nModel ready for deployment.")
print("Can be used to forecast daily demand based on weather & calendar features.")
```

## Next Options (Let me know if needed):

-  Add plots (Actual vs Predicted)
-  Add Random Forest / Gradient Boosting
-  Feature importance analysis
-  Export model with joblib (.pkl)

Would you like a **.py file download** or **Jupyter Notebook (.ipynb)** version of this script?



Downloading from [https://www.kaggle.com/api/v1/datasets/download/gauravduttakiit/bike-sharing?dataset\\_version\\_number=1...](https://www.kaggle.com/api/v1/datasets/download/gauravduttakiit/bike-sharing?dataset_version_number=1...)  
 100% |  | 21.1k/21.1k [00:00<00:00, 23.4MB/s]Extracting files...  
 Path to dataset files: /root/.cache/kagglehub/datasets/gauravduttakiit/bike-sharing/versions/1

Great! It looks like you've successfully downloaded the **Bike-Sharing dataset** from Kaggle, and it's located at:

```
swift
```

```
/root/.cache/kagglehub/datasets/gauravduttakiit/bike-sharing/versions/1
```

Now, you can directly load the dataset in your script using that path.

## Next Step — Load the CSV File

Inside that path, look for a file like `day.csv` or `hour.csv`.  
Most Kaggle bike-sharing datasets use one of those names.  
Here's how to load it in Python:

```
python

import pandas as pd

# Update the path based on the exact filename
file_path = "/root/.cache/kagglehub/datasets/gauravduttakiit/bike-sharing/versions/1/day.csv"
# or
# file_path = "/root/.cache/kagglehub/datasets/gauravduttakiit/bike-sharing/versions/1/hour.csv"

df = pd.read_csv(file_path)
print(df.head())
```

✓ **Want me to update the full CRISP-DM script using this exact dataset path?**

I can generate a **ready-to-run Python file** with:

- Correct file path included
- Full preprocessing + model
- Outlier analysis
- Optional visualizations (Actual vs Predicted)

Reply with one of these:

- 🟢 **"Full script ready with this dataset"** → I'll create the complete `.py` file.
- 🔍 **"Use day.csv or hour.csv?"** → If you're unsure which file to model.



以下是作業目標 幫我確認你給我的程式碼是否滿足第一點到第三點

一、作業目標：

本次作業延伸自 HW1，目標是讓同學能夠以實際資料集進行「多元線性回歸 (Multiple Linear Regression)」的完整分析，並遵循 CRISP-DM 流程完成從資料理解、建模到評估的全過程。

二、作業內容：

1. 資料來源

至 Kaggle 選擇一個具有 10 至 20 個特徵 (features) 的公開資料集。

類型不限（可為房價預測、醫療、車輛效能等主題）。

請明確標示資料集來源與連結。

2. 分析任務

使用線性回歸 (Linear Regression) 模型進行預測。

可嘗試單純線性回歸、多元線性回歸或 Auto Regression。

必須執行 特徵選擇 (Feature Selection) 與 模型評估 (Model Evaluation)。

結果部分需包含請提供預測圖(加上信賴區間或預測區間)

### 3. CRISP-DM 流程說明

Business Understanding  
Data Understanding  
Data Preparation  
Modeling  
Evaluation  
Deployment

### 4. AI協助要求

所有與 ChatGPT 的對話請以 pdfCrowd 或其他方式匯出為 PDF

請使用 NotebookLM 對網路上同主題的解法進行研究，並撰寫一份 100 字以上的摘要，放入報告中。

請在報告中明確標示「GPT 輔助內容」與「NotebookLM 摘要」

### 5. 繳交內容

主程式：7114056XXX\_hw2.py/.ipynb

報告檔：PDF，需包含以下內容：

按照 CRISP-DM 說明的分析流程

GPT 對話過程（pdfCrowd 匯出）

NotebookLM 研究摘要

網路上主流或更優解法之比較與說明

以上檔案與資料夾請壓縮為學號命名的一個zip（例如 7114056XXX\_hw2.zip）上傳。

(optional) 若上傳至 GitHub，或是以 colab 撰寫，需提供連結，並在 README.md 中整理流程與成果。

### 三、評分標準

文件說明（50%）

CRISP-DM 流程完整且邏輯清楚（25%）

包含 GPT 對話與 NotebookLM 摘要（15%）

有明確說明資料集來源與研究脈絡（10%）

結果呈現（50%）

模型正確可執行，具特徵選擇與評估（25%）

結果合理、美觀且具有說服力（15%）

呈現出 Kaggle 名次(若有)/預測結果評估(預測圖、評估指標)（10%）

好的，我來協助你確認目前我提供的 **Bike Sharing + Linear Regression 程式碼** 是否有滿足作業要求中的 **第 1~3 點**。

## ✓ 核對作業要求 vs 我提供的內容

### (1) 資料來源要求：Kaggle + 10~20 個特徵

要求	是否符合？	說明
Kaggle 公開資料集	✓ 可以符合	Bike Sharing Dataset 來自 Kaggle（需在報告中附上 Kaggle 連結）

要求	是否符合？	說明
10-20 個特徵	⚠ 需確認 & 擴充	<code>day.csv</code> 大約 12 個特徵，但目前我程式碼有刪除多個欄位（如 <code>casual</code> , <code>registered</code> 、日期欄位）。→ 需保留足夠特徵並明確列出欄位

✅ 要求 1 可以符合，但需要在報告中加入 Kaggle 連結並保留完整特徵欄位。

## (2) 分析任務：線性回歸 + 特徵選擇 + 評估 + 預測圖

要求	是否符合？	說明
使用線性回歸模型	✅ 已有	使用 <code>LinearRegression()</code>
特徵選擇 (Feature Selection)	❌ 尚未提供	我提供的程式沒有做特徵選擇（如 VIF、p-value、RFE、Lasso 等）
模型評估指標	✅ 部分	已有 RMSE, $R^2$
預測圖 (必須有)	❌ 尚未提供	需加入「實際 vs 預測圖」+ 信賴區間或預測區間

❗ 第 2 點目前「不完全符合」，需要加入：

- 特徵選擇
- 預測圖與信賴區間
- 可選：加入更佳的模型比較（如 Ridge/Lasso）

## (3) CRISP-DM 流程

要求	是否符合？	說明
Business Understanding	✅ 有	
Data Understanding	✅ 有 (head/info)	
Data Preparation	✅ 有 (OneHot + Scaling)	
Modeling	✅ 有 (Linear Regression)	
Evaluation	✅ 部分 (指標有，但缺圖 & 說明)	
Deployment	✅ 有簡述	

✅ CRISP-DM 架構基本有，但為符合作業 文件評分 25%，還需寫敘述文字（非只有程式碼）。

## 🎯 結論：目前程式碼符合度

作業項目	狀態
1. Kaggle 資料集	✅ 可符合
2. 線性回歸 + 評估	⚠ 尚不足
特徵選擇	❌ 缺少
預測圖 + 信賴區間	❌ 缺少
3. CRISP-DM 架構	✅ 基本具備（仍需文字敘述）



## 🔧 下一步我可以幫你補上：

- 🔍 Feature Selection (RFE/Lasso/VIF)
- 📊 預測圖 + 信賴區間 (Regression plot)
- 📄 CRISP-DM 說明文字 (可以放入報告)
- 🔗 Kaggle 資料集來源標註格式
- 🖌️ Bonus: Ridge、Lasso、Feature Importance

## ▶▶ 請回答：

要我幫你升級程式碼以完全符合作業第 1~3 點嗎？

回覆：

- 🟢 「是，幫我補完整特徵選擇 + 預測圖」
- 🔵 「給我可以直接交的 .py 或 .ipynb 檔」
- 🟡 「先補 Feature Selection 就好」

我會幫你變成 可直接繳交的作業框架。



- 🟢 「是，幫我補完整特徵選擇 + 預測圖」

太好了！我會幫你在原本的 Bike Sharing 線性回歸專案中加入兩個關鍵功能，讓你的作業完全符合第 2 點的要求：

- ✓ Feature Selection (特徵選擇)
- ✓ 預測圖 (含信賴區間或誤差帶)

## 🔧 我接下來會新增的內容

功能	方法
🖌️ 特徵選擇 (Feature Selection)	使用 <b>RFE (Recursive Feature Elimination)</b> 或 <b>LassoCV</b> ，自動挑出最重要特徵
📊 預測圖	回歸散佈圖：實際值 vs 預測值
📊 信賴區間或誤差區間	使用 <b>標準差 ± 預測誤差</b> 或 <b>Bootstrapping</b>

## 📁 最終程式架構 (升級版)

- CRISP-DM 架構完整
- 含模型訓練、檢驗、特徵重要性、殘差分析、預測圖
- 可直接匯出 .py 或 .ipynb

## 確認一下兩件事（請回答數字）：

### 1 用哪一個資料表？

- `day.csv`（每日資料）
- `hour.csv`（每小時資料）

### 2 模型種類是否只用 Linear Regression？

- a. 只要 **Linear Regression**
- b. 想額外比較 **Lasso/Ridge**（可加分）

 回覆格式示例：

1. `day.csv`    2. b. 加 **Lasso/Ridge**

我確認後就會開始製作 **完整版可交作業程式碼**（含圖 & Feature Selection）。



1. `day.csv` 2. a


了解 

我們將使用：


- **資料集**：`day.csv`（每日租借資料）
- **模型**：僅使用 **Linear Regression**（不加入 Lasso/Ridge）

我會為你建立一個完全符合作業要求的版本，包含：

## 將新增的功能



CRISP-DM 階段	要新增的內容
Data Preparation	保留足夠特徵（10~20）並說明欄位
Feature Selection	使用 <b>RFE (Recursive Feature Elimination)</b> 找出重要特徵
Modeling	線性回歸模型
Evaluation	RMSE、 $R^2$ 、殘差分析
Visualization	 預測圖（Actual vs Predicted） + 信賴區間/誤差帶
Outlier Analysis	Top 5 殘差最大點


## 模型視覺化：將製作的圖

- 1 實際 vs 預測 圖 (Regression Scatter Plot)
- 2 含信賴區間 / 誤差區間
- 3 殘差分佈圖 (如果需要可額外加 )

## 程式輸出格式

你想要最後是：

-  `.py` 檔 (Python 腳本) 或
-  `.ipynb` (Jupyter Notebook 可附在作業報告)

 最後請告訴我一件事 (回覆 A 或 B)：

- A ☐ 我需要 `.py` 可直接執行
- B ☐ 我需要 `.ipynb` 有註解、方便寫報告

我收到後，開始製作最終作業版本 (含中文 CRISP-DM 說明架構可直接放報告)。



B

Done — I created the Jupyter Notebook `.ipynb` for your HW2 and saved it here:

[Download the notebook](#)

What I included:

- Full CRISP-DM structured notebook (Business → Deployment) with markdown and runnable code cells.
- Data Preparation, RFE feature selection (top 10), Linear Regression modeling, evaluation (RMSE,  $R^2$ ), residuals, and prediction-interval calculations.
- Plots: Actual vs Predicted and Actual/Predicted with 95% prediction intervals (cells included — they will run once you set `DATA_PATH` to your environment's `day.csv` path and execute the notebook).
- Saved helper outputs: instructions on where to export figures/CSV for your report.

Next steps for you:

1. Open the notebook in Jupyter / Colab / VS Code.
2. Edit the `DATA_PATH` variable in the first code cell to the actual path on your environment if different (example: the Kaggle path you shared).
3. Run cells top→bottom. The notebook will produce the figures and CSVs you can include in your PDF report.

If you want, I can now:

- Run the notebook here if you give me a valid path to `day.csv` in this environment (I attempted earlier but the file path wasn't present).
- Or, customize the notebook further (add more commentary in Chinese tailored for your report, include code to export figures/CSV automatically, or produce the PDF-ready report layout).