1、基于生成数据、多项式特征生成，数据归一化和SVM构建机器学习流水线

（1）使用make\_classification 或者 make\_moons()生成二维二元分类数据集，根据标签进行样本数据可视化

（2）参数：多项式特征生成PolynomialFeatures()， 归一化方式StandardScaler()，

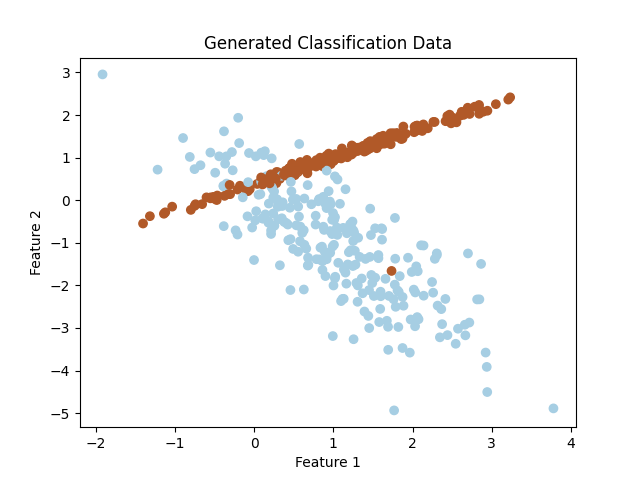
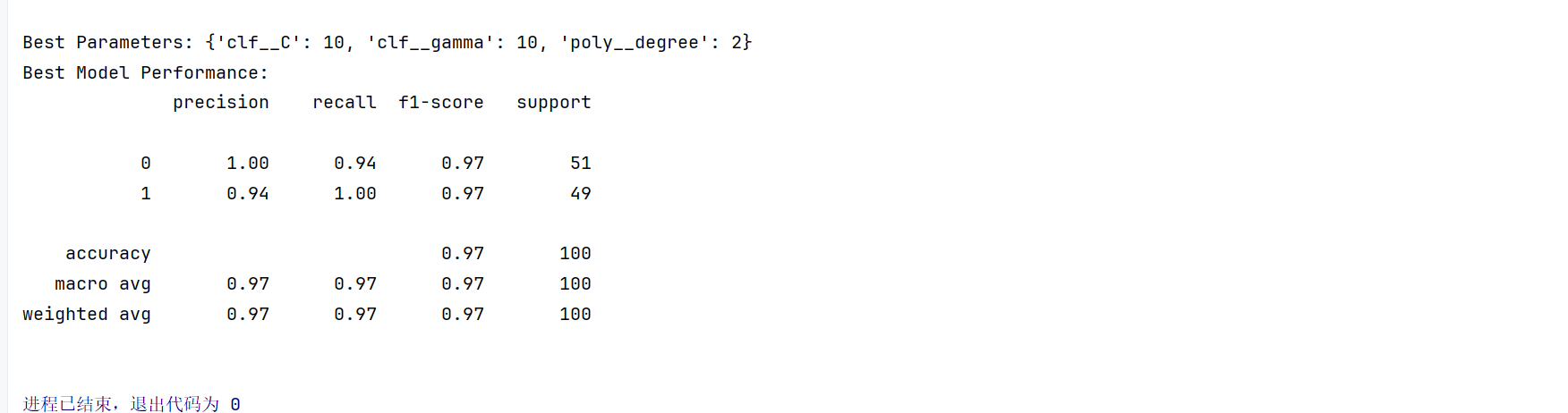
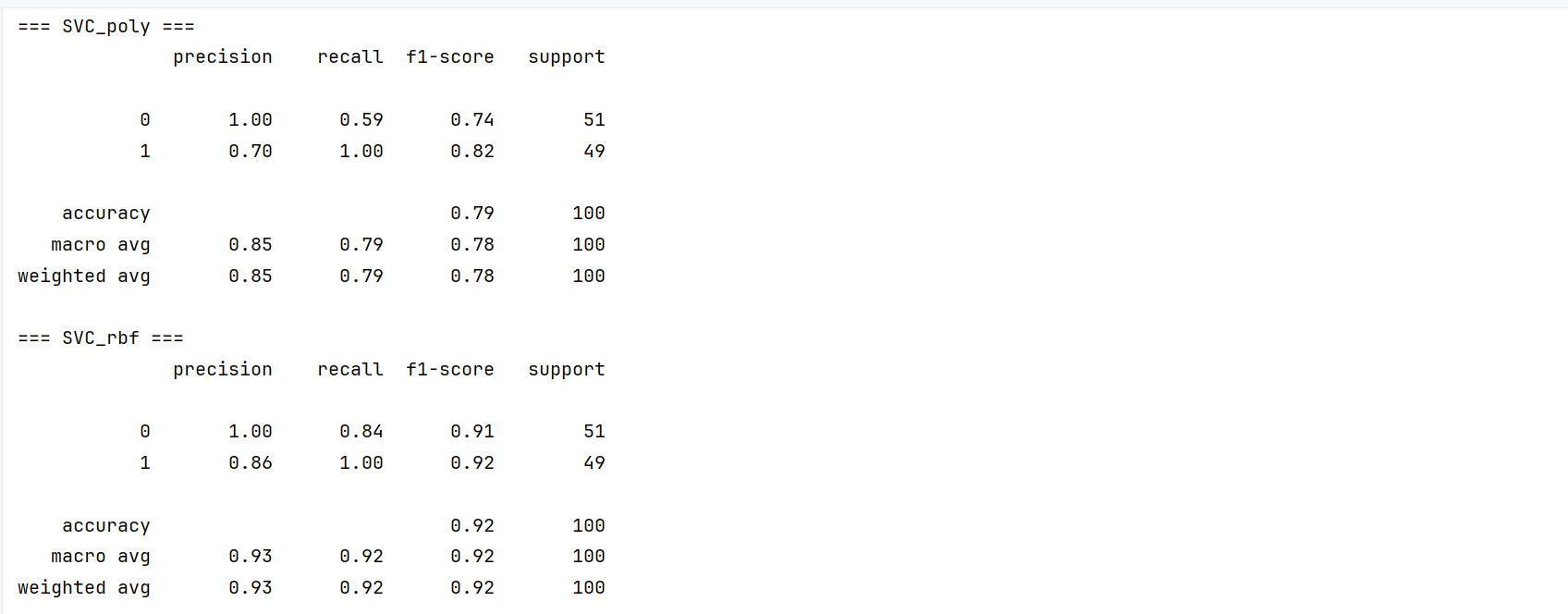
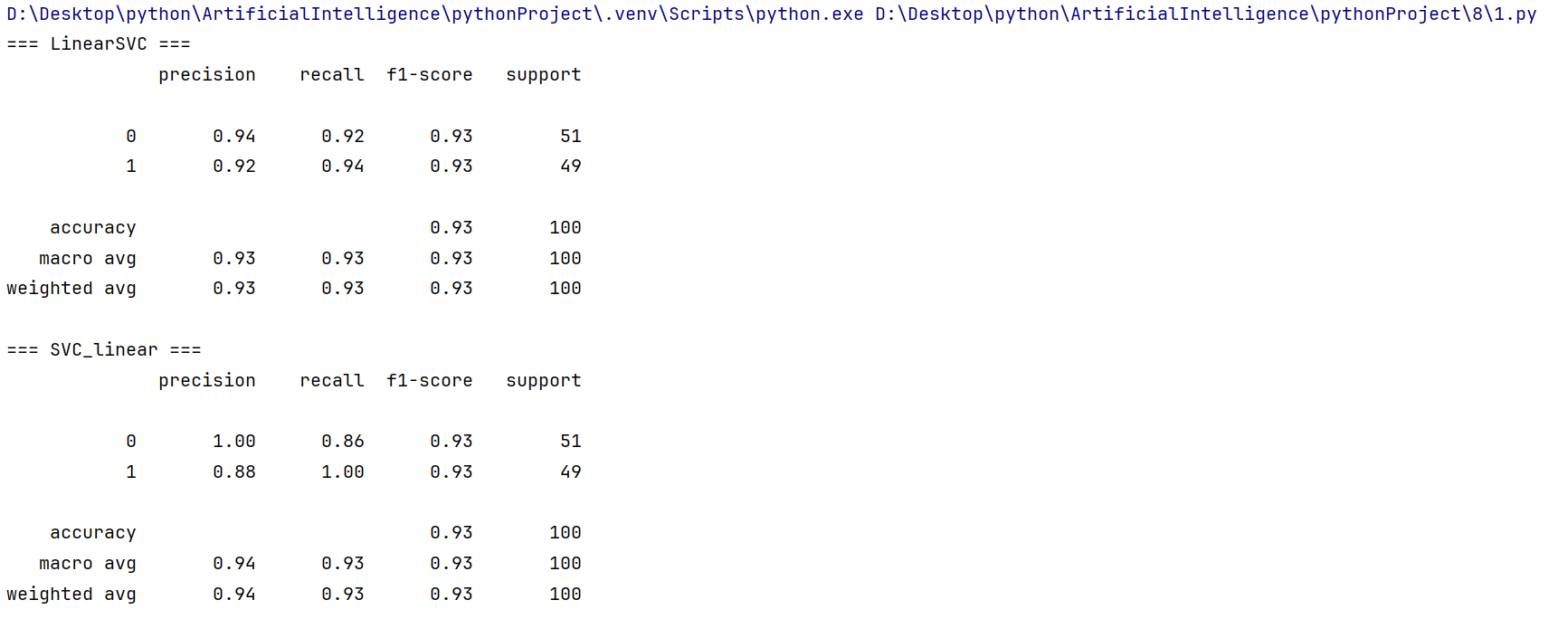
SVM的选择：线性 LinearSVC()和SVC()

多项式 SVC()和SVC()

构建机器学习流水线模型Pipeline()，基于流水线模型训练输出各种参数下，流水线模型对应的性能报告

（3）将网格调参GridSearchCV与Pipeline进行结合使用构建基于流水线的网格寻参模型，进行训练数据，输出最优参数集合和最优模型，使用最优模型测试模型的测试集输出测试集的性能报告

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.datasets import make\_classification  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import PolynomialFeatures, StandardScaler  
from sklearn.svm import LinearSVC, SVC  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.metrics import classification\_report  
  
# 生成三维二元分类数据集并可视化  
X, y = make\_classification(n\_samples=500, n\_features=2, n\_classes=2, n\_clusters\_per\_class=1,  
 n\_informative=2, n\_redundant=0, n\_repeated=0, random\_state=42)  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.title('Generated Classification Data')  
plt.show()  
  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 构建机器学习流水线  
pipelines = {  
 'LinearSVC': Pipeline([('poly', PolynomialFeatures()), ('scaler', StandardScaler()), ('clf', LinearSVC(dual='auto'))]),  
 'SVC\_linear': Pipeline([('poly', PolynomialFeatures()), ('scaler', StandardScaler()), ('clf', SVC(kernel='linear'))]),  
 'SVC\_poly': Pipeline([('poly', PolynomialFeatures()), ('scaler', StandardScaler()), ('clf', SVC(kernel='poly'))]),  
 'SVC\_rbf': Pipeline([('poly', PolynomialFeatures()), ('scaler', StandardScaler()), ('clf', SVC(kernel='rbf'))])  
}  
  
# 训练模型并输出性能报告  
for name, pipeline in pipelines.items():  
 pipeline.fit(X\_train, y\_train)  
 y\_pred = pipeline.predict(X\_test)  
 print(f"=== {name} ===")  
 print(classification\_report(y\_test, y\_pred))  
  
# 结合网格调参寻找最优参数组合  
parameters = {  
 'poly\_\_degree': [2, 3],  
 'clf\_\_C': [0.1, 1, 10],  
 'clf\_\_gamma': [0.1, 1, 10]  
}  
  
grid\_search = GridSearchCV(pipelines['SVC\_rbf'], parameters, cv=5, n\_jobs=-1)  
grid\_search.fit(X\_train, y\_train)  
  
# 输出最优参数组合和最优模型的性能报告  
best\_params = grid\_search.best\_params\_  
best\_model = grid\_search.best\_estimator\_  
print("Best Parameters:", best\_params)  
y\_pred = best\_model.predict(X\_test)  
print("Best Model Performance:")  
print(classification\_report(y\_test, y\_pred))



2、多特征分类中的特征选择

（1）导入datasets.load\_digit()数据，使用 SelectKBest(chi2, k>=20)选择最优特征

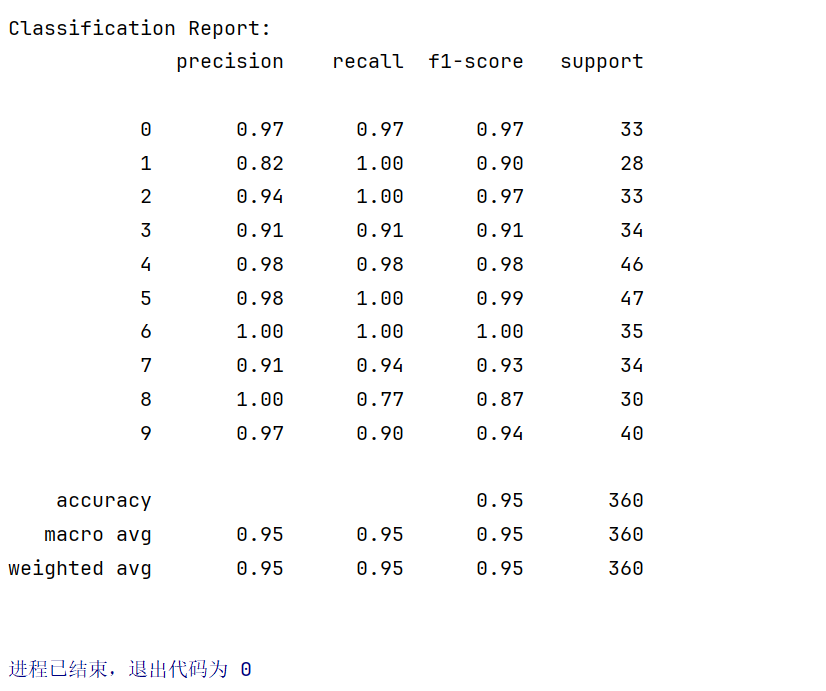
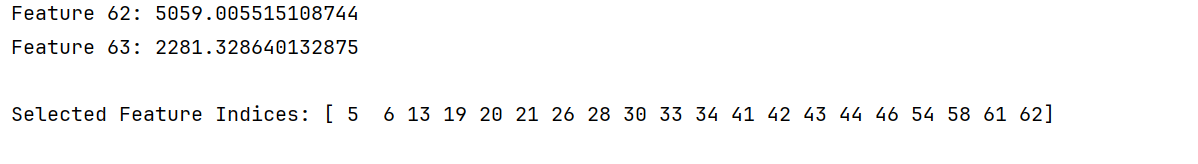
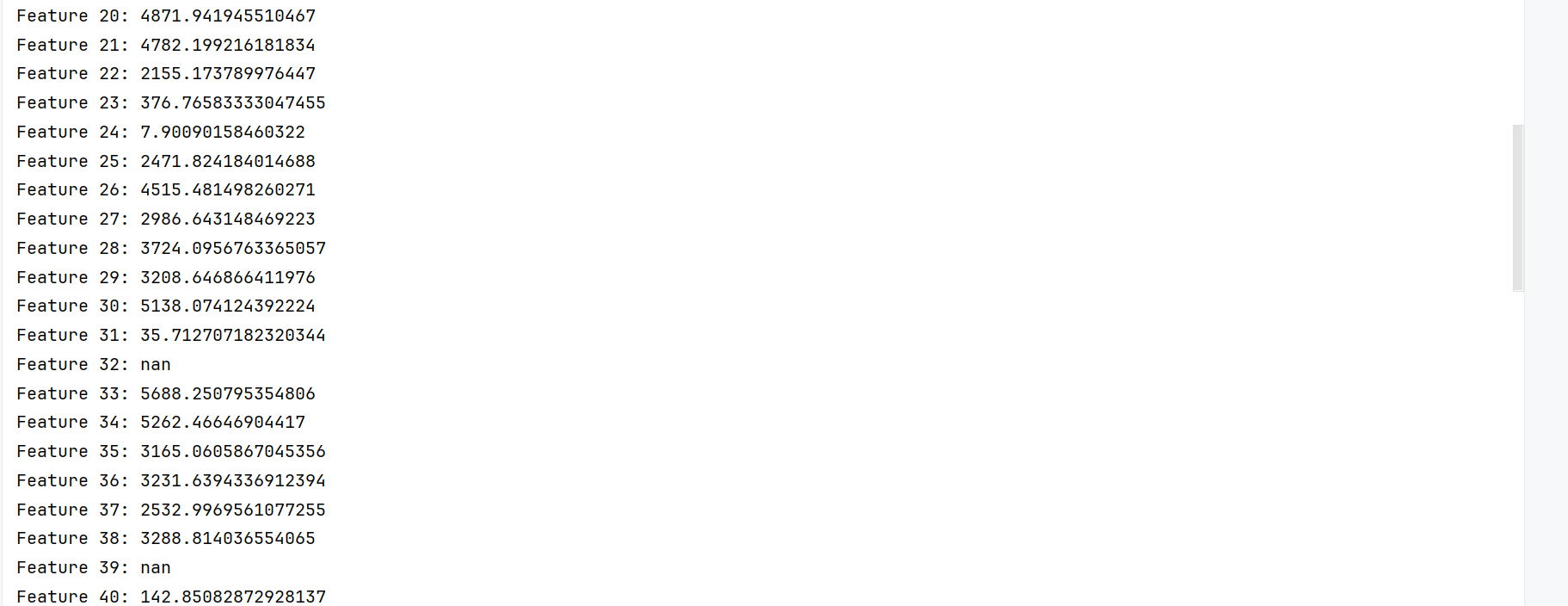
（2）输出各个特征的的得分和返回特征过滤后保留下的特征列索引

（3）基于特征过滤后保留下的特征数据集和SVC（参数自己定）构建机器学习流水线

Pipeline([('selector', ),

('svm', )])，使用该模型训练数据，输出性能报告

from sklearn.datasets import load\_digits  
from sklearn.feature\_selection import SelectKBest, chi2  
from sklearn.pipeline import Pipeline  
from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
  
# 导入数据集  
digits = load\_digits()  
X, y = digits.data, digits.target  
  
# 特征选择  
selector = SelectKBest(chi2, k=20) # 选择得分最高的20个特征  
# X\_selected = selector.fit\_transform(X, y)  
selector.fit(X, y)  
selected\_feature\_indices = selector.get\_support(indices=True)  
feature\_scores = selector.scores\_  
  
# 输出特征得分和选择后的特征列索引  
print("Feature Scores:")  
for i, score in enumerate(feature\_scores):  
 print(f"Feature {i}: {score}")  
print("\nSelected Feature Indices:", selected\_feature\_indices)  
  
# 划分训练集和测试集  
# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
# 构建机器学习流水线  
pipeline = Pipeline([  
 ("selector",SelectKBest(chi2, k=20)),  
 ('svm', SVC(kernel='rbf', C=1.0, gamma='scale')) # 参数可以自行调整  
])  
  
# 训练模型并输出性能报告  
pipeline.fit(X\_train, y\_train)  
y\_pred = pipeline.predict(X\_test)  
print("\nClassification Report:")  
print(classification\_report(y\_test, y\_pred))



3、多特征选择和SVM回归SVR的结合

（1）导入datasets.fetch\_california\_housing()数据，使用SelectKBest(f\_regression, k=4)选择最优特征

（2）输出各个特征的的得分和返回特征过滤后保留下的特征列索引,4个特征的名称

（3）基于特征过滤后保留下的特征数据集和回归SVR(参数自己定)

构建机器学习流水线Pipeline([('selector', ), ('svm', )])，使用该模型训练数据，输出MSE和MAE

from sklearn.datasets import fetch\_california\_housing  
from sklearn.feature\_selection import SelectKBest, f\_regression  
from sklearn.pipeline import Pipeline  
from sklearn.svm import SVR  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error  
  
# 导入数据集  
housing = fetch\_california\_housing()  
X, y = housing.data, housing.target  
# print(X.shape)  
# 特征选择  
selector = SelectKBest(f\_regression, k=4) # 选择与目标变量最相关的 10 个特征  
selector.fit(X, y)  
selected\_feature\_indices = selector.get\_support(indices=True)  
feature\_scores = selector.scores\_  
  
# 输出特征得分和选择后的特征列索引  
print("Feature Scores:")  
for i, score in enumerate(feature\_scores):  
 print(f"Feature {i}: {score}")  
print("\nSelected Feature Indices:", selected\_feature\_indices)  
  
# 获取特征名称  
selected\_feature\_names = [housing.feature\_names[i] for i in selected\_feature\_indices]  
print("\nSelected Feature Names:", selected\_feature\_names)  
  
  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 构建机器学习流水线  
pipeline = Pipeline([  
 ("selector",SelectKBest(f\_regression, k=4)),  
 ('svm', SVR(kernel='rbf', C=1.0, gamma='scale')) # 参数可以自行调整  
])  
  
# 训练模型并预测  
pipeline.fit(X\_train, y\_train)  
y\_pred = pipeline.predict(X\_test)  
  
# 计算 MSE 和 MAE  
mse = mean\_squared\_error(y\_test, y\_pred)  
mae = mean\_absolute\_error(y\_test, y\_pred)  
print("\nMean Squared Error (MSE):", mse)  
print("Mean Absolute Error (MAE):", mae)

