《**数据挖掘技术**》实验考核表

实验名称： 实验一：数据挖掘算法（1）-数据回归与预测

2020级 专业： 计算机科学与技术

实验时间： 2023年4月6日 3-4节中午 实验地点： 7-212

表一

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 班级 | 实验摘要与总结（学生填写） | 成绩 |
| **2020** |  | 计算机8班 | 在本实验中，我实现了sklearn库中四种多项式拟合方法的使用。解决了许多导致拟合结果异常的难题。实现了交叉验证。并在未来数据的预测上实现了较高的精确度。 |  |

表二

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 考核标准 | 实验过程  30分 | 实验报告  30分 | 程序运行  20分 | 交流合作  10分 | 考勤纪律  10分 |
| 各项成绩 |  |  |  |  |  |
| 考核内容 | 评价实验课堂中的综合表现，包括实验态度、实验过程等内容。 | ○完整  ○较完整  ○一般  ○内容极少 | ○运行正常  ○功能不全  ○有小错  ○无法运行 | ○良好  ○一般  ○较差 | ○良好  ○一般  ○较差 |

表三

|  |
| --- |
| 其它批改意见 |
|  |

注：

* 表1除“成绩”栏外由学生填写，表2、表3由指导教师填写。
* 本页附在实验报告正文前装订，纸质版报告可单双面打印，**电子报告Word排版**。
* **实验报告如发现雷同，一律以0分计。**

**一、实验目的**

1. 理解数据回归的基本概念；
2. 掌握使用最小二乘法进行数据拟合，掌握最小二乘法的计算流程；
3. 掌握使用sklearn进行数据拟合；
4. 能够根据回归情况，对数据进行预测；能够分析数据回归的情况。
   1. **实验内容**
5. 准备数据。数据网站：https://q.stock.sohu.com/zs/000001/lshq.shtml
6. 数据预处理：选取 “close” 列数据为**拟合**对象，自行确定**拟合**和预测使用的数据长度，归一化等技术；
7. 使用最小二乘法的不同**多项式参数**分别拟合该曲线，**参数**的选择要尽可能多样性，对比不少于**3组的不同参数**，绘制**拟合曲线**，测试**MSE误差**，选择较为优化的参数；
8. 以（2）中优化参数为模型，预测未来5日的数据；
9. 绘制**流程图**，描述**最小二乘法**的计算过程；使用python实现**最小二乘法**，并测试；
10. 使用Ridge回归/岭回归、Lasso回归、ElasticNet回归，预测未来5日数据，并计算误差率。
11. 5日后根据实际情况，填写表格，并计算误差率。
12. （选做）使用KNN、SVR、AdaBoost预测方法，预测未来5日数据，并计算误差率。

**三、实验要求**

实验报告中体现全部实验内容，至少包括：

1. 数据拟合程序关键代码；
2. 拟合曲线图；
3. 未来五日的预测数据、实际数据（五日后填写）和预测误差百分比，预测误差 =（预测值-真实值）/真实值。
4. 实验结果的分析；

**四、实验步骤**

（以下部分按实验要求填写，序号对应实验内容，此部分只可粘贴核心代码。）

1.

# 这部分是demo1.1的导入数据

df = pd.read\_csv('C:/Users/24957/Desktop/指数.csv')

data = df["close"].values

data = data[6883:7883]

# 使用归一化准备X和Y的数据，这里选用了sklearn的归一化，y选用了后一千个数据

X = np.array([i for i in range(0, len(data) + 5)]).reshape(-1,1)

scaler = MinMaxScaler()

ALL = scaler.fit\_transform(X)

X = ALL[:1000]

X\_predict = ALL[1000:]

Y = data

2.

degrees = [1, 5, 10, 17]

for i in degrees:

# 老三样

linear\_regression = LinearRegression()

polynomial\_features = PolynomialFeatures(i, include\_bias=False)

pipeline = Pipeline([("polynomial\_features", polynomial\_features), ("linear\_regression", linear\_regression)])

pipeline.fit(X, Y)

# 通过交叉验证进行评分

scores = cross\_val\_score(pipeline, X, Y, scoring="neg\_mean\_squared\_error", cv=10)

mean\_neg\_mse = np.mean(scores)

mean\_mse = -mean\_neg\_mse

print("平均均方误差\_Linear:", mean\_mse)

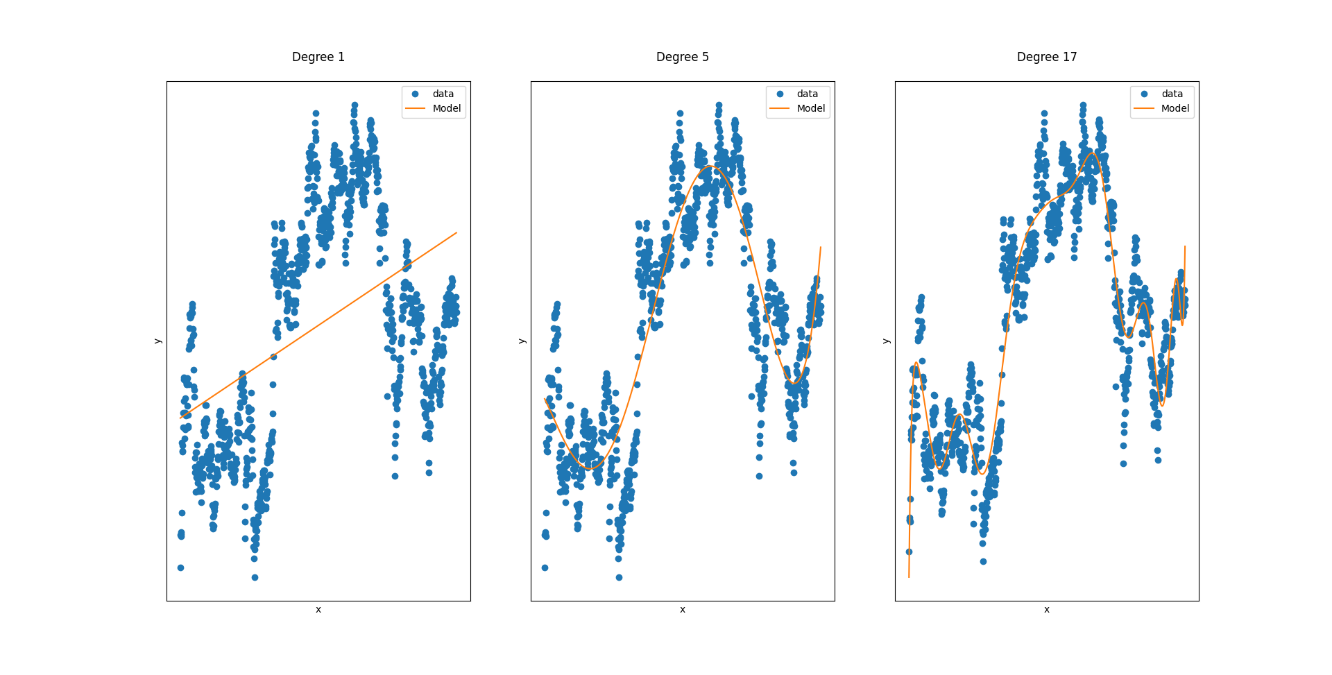
# 画图

3. 最小二乘回归实验结果分析：

print(pipeline.predict(ALL[1000:]))

**经过测试，此实验结果选择的是拟合度最好的参数为16的多项式**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序列 | 第1日 | 第2日 | 第3日 | 第4日 | 第5日 |
| 日期 | 3月29日 | 3月30日 | 3月31日 | 4月3日 | 4月4日 |
| 实际值 | 3240.057 | 3261.249 | 3272.86 | 3296.40 | 3312.56 |
| 预测值 | 3265.01366504 | 3272.67382129 | 3282.54100879 | 3294.89843066 | 3310.05272754 |
| 预测误差 | 0.77% | 0.34% | 0.31% | 0.06% | 0.06% |



4. import numpy as np

x = np.array([0, 1, 2, 3, 4])

y = np.array([1, 3, 5, 7, 10])

# 计算的平均值

x\_mean = np.mean(x)

y\_mean = np.mean(y)

# 计算差值

x\_diff = x - x\_mean

y\_diff = y - y\_mean

# 计算斜率和截距

k = np.sum(x\_diff \* y\_diff) / np.sum(x\_diff\*\*2)

b = y\_mean - k \* x\_mean

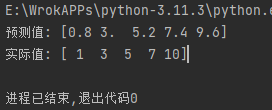
# 测试拟合直线

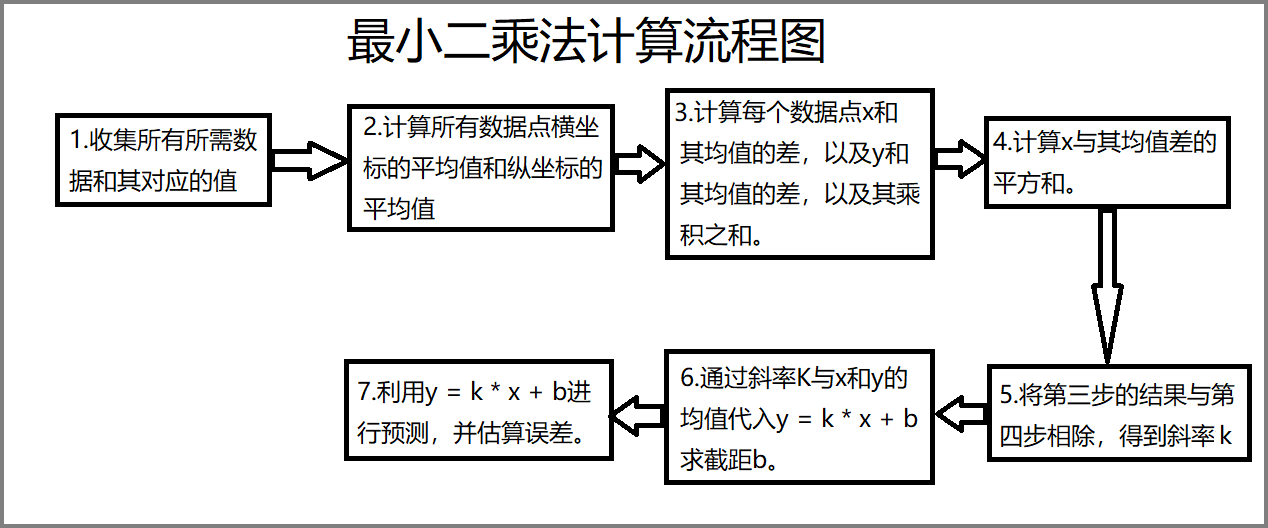
y\_pred = k \* x + b

print("预测值:", y\_pred)

print("实际值:", y)

预测结果：





1. Ridge回归、Lasso回归、ElasticNet回归结果分析

ridge = Ridge(alpha=0.01)

polynomial\_features\_Ridge = PolynomialFeatures(18, include\_bias=False)

pipeline\_Ridge = Pipeline([("polynomial\_features", polynomial\_features\_Ridge), ("ridge", ridge)])

pipeline\_Ridge.fit(X,Y)

# 预测未来五个数据

y\_pred = pipeline\_Ridge.predict(X\_predict)

lasso = Lasso(alpha=0.5, max\_iter=19000)

polynomial\_features\_Lasso = PolynomialFeatures(10, include\_bias=False)

pipeline\_Lasso = Pipeline([("polynomial\_features", polynomial\_features\_Lasso), ("lasso", lasso)])

pipeline\_Lasso.fit(X,Y)

# 预测未来五个数据

y\_pred = pipeline\_Lasso.predict(X\_predict)

elasticNet = ElasticNet(alpha=0.01)

polynomial\_features\_ElasticNet = PolynomialFeatures(20, include\_bias=False)

pipeline\_ElasticNet = Pipeline([("polynomial\_features", polynomial\_features\_ElasticNet), ("elasticNet", elasticNet)])

pipeline\_ElasticNet.fit(X,Y)

# 预测未来五个数据

y\_pred = pipeline\_ElasticNet.predict(X\_predict)

6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序列 | 第1日 | 第2日 | 第3日 | 第4日 | 第5日 |
| 日期 | 3月29日 | 3月30日 | 3月31日 | 4月3日 | 4月4日 |
| 实际值 | 3240.057 | 3261.249 | 3272.86 | 3296.40 | 3312.56 |
| Ridge预测值 | 3294.1630833 | 3294.6155103 | 3294.9242584 | 3295.0835429 | 3295.0874504 |
| Ridge预测误差 | 1.67% | 1.01% | 0.67% | 0.03% | 0.51% |
| Lasso预测值 | 3286.4724581 | 3291.867305 | 3297.3779038 | 3303.0055625 | 3308.751602 |
| Ridge预测误差 | 1.42% | 0.92% | 0.76% | 0.21% | 0.12% |
| ElasticNet预测值 | 3293.6644318 | 3300.654289 | 3307.8554735 | 3315.2720373 | 3322.9080995 |
| ElasticNet预测误差 | 1.63% | 1.20% | 1.07% | 0.58% | 0.30% |

7.（选做）参考5的表格

**五、实验总结**

（实验结果分析、收获、问题、解决方法、最优方案、建议体会等…）

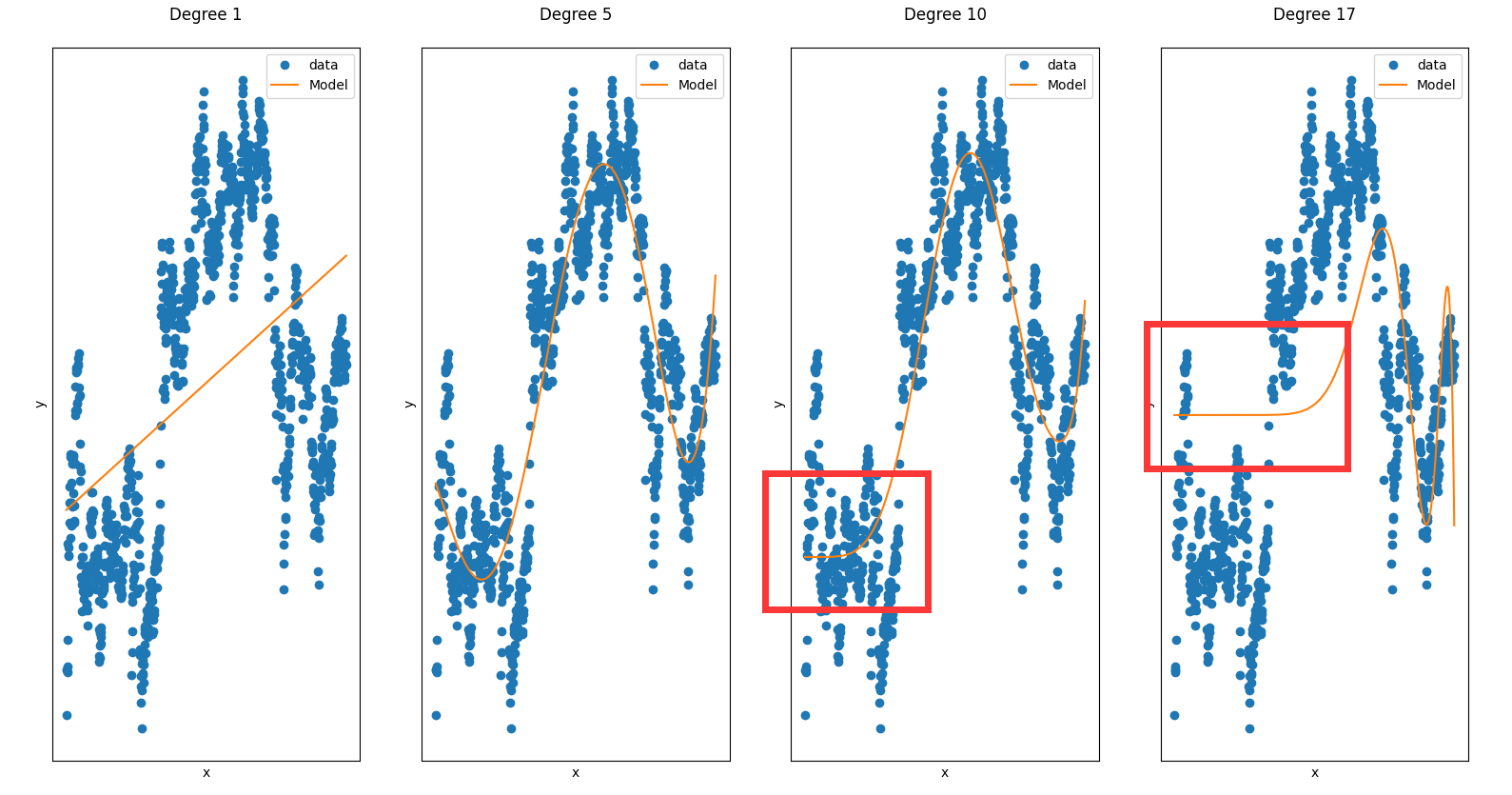
本次实验中，我首先使用sklearn库中的OLS方法实现了对一千条数据的多项式拟合。在进行拟合的过程中，我选用了1、5、17的多项式参数。由于实验的数据量较大，为了避免多项式高阶项对于拟合结果的影响过大，我选择了sklearn库中的归一化函数 — MinMaxScaler()对输入的X进行了归一化处理。

（一）

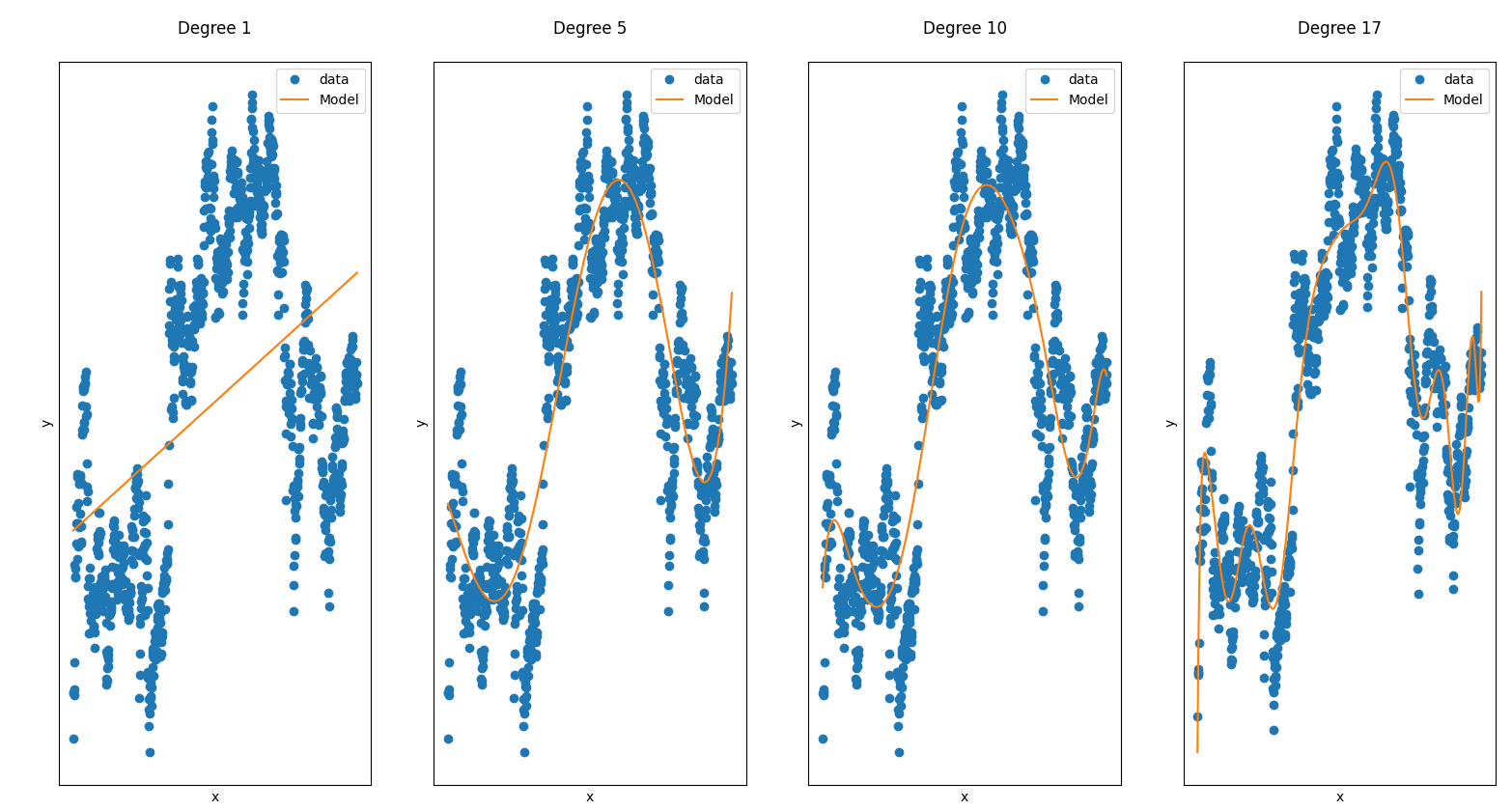
在这其中遇到的第一个问题是：与numpy的简单直接不同，为什么sklearn的多项式拟合需要通过管道，将PolynomialFeatures与LinearRegression相连后才能进行？

对于这个问题，通过查阅资料我得知，sklearn并不存在一个专门的多项式拟合函数，其LinearRegession()方法中所包含的只是一个最基本的OLS算法。而其对多项式的拟合是通过PolynomialFeatures方法对于输入数据进行预处理，以模拟多项式拟合的输入进行的。

而这就导致了一个严重的问题——在numpy中存在专门的多项式拟合参数np.polyfit，其针对多项式拟合具有相较于sklearn更强的健壮性。它对于输入数据的范围是不敏感的。而sklearn所使用的原始最小二乘法的问题在于——它对输入数据X的分布过于敏感了，这会导致在输入数据的范围为0-1000的情况下的一个严重后果，假如不对数据进行归一化处理，那么它的高阶项对于拟合结果的影响会无限膨胀，以至于让拟合模型在低阶项上的权重分布趋于0，最终呈现出随着多项式参数的升高，其在拟合图像的前半部分趋于直线的情况：



而在对X进行归一化后，拟合结果趋于正常：

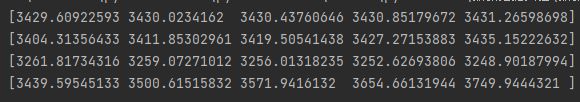


然而，归一化也存在着自己的问题，由于归一化后的X与原本的X值完全不同，因此对于之后五天数据的预测，例如对第1001天的cost就不能仅仅使用fit.predict([[1001]])这种写法。我需要首先创建一个长度为1005的列表，归一化后，将前1000个值赋予X以用作拟合，而将后5个值切片后用于预测，最终才得到正确的预测值。

从这里就可以看出，sklearn的线性回归在具有更高的灵活性的同时，使用门槛也大大增加了。

（二）

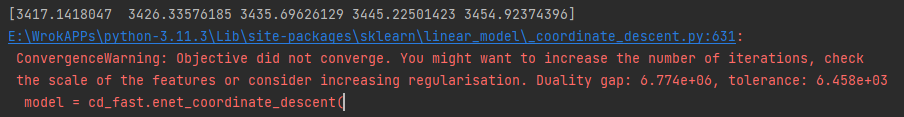
之后，在尝试使用其他三种模型进行预测时，我发现了一个令人吃惊的事实，以下是参数为1、5、10、17的线性回归模型对于后五天的预测结果：

从中可以看到，即使是直线模型，其预测误差率也在5%以下，甚至比参数为17的模型表现还要好。关于这一事实，我的推测是，源数据的分布本身就较为集中与3400左右的区域，而由于Y的“基准值”3400过大，稀释了误差率。

可以直接推论，这种对于误差率的计算方式是不科学的，不能在不同分布的数据之间比较模型的拟合能力，也很难说反映了模型的真实能力，但是由于本实验的限制，此问题暂且搁置。

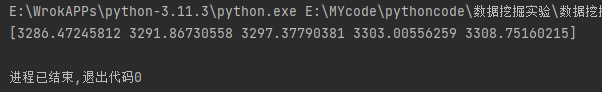
（三）

之后，在尝试使用Lasso、Ridge、elasticNet三种模型进行拟合时，都出现了不同程度的收敛报错，这表明模型没有成功收敛，需要通过尝试增大迭代次数，或者调整正则化参数alpha来解决问题：

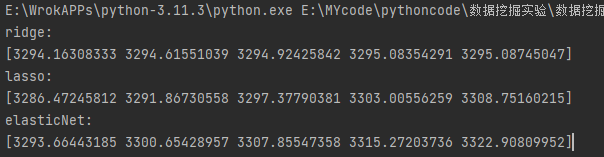


内置函数的默认迭代次数为1000。经过反复测试，在这里尝试将其迭代次数设置为8000，正则化参数设置为0.5时，取得了稳定的收敛，并在多项式参数为10时取得了较好的拟合结果：

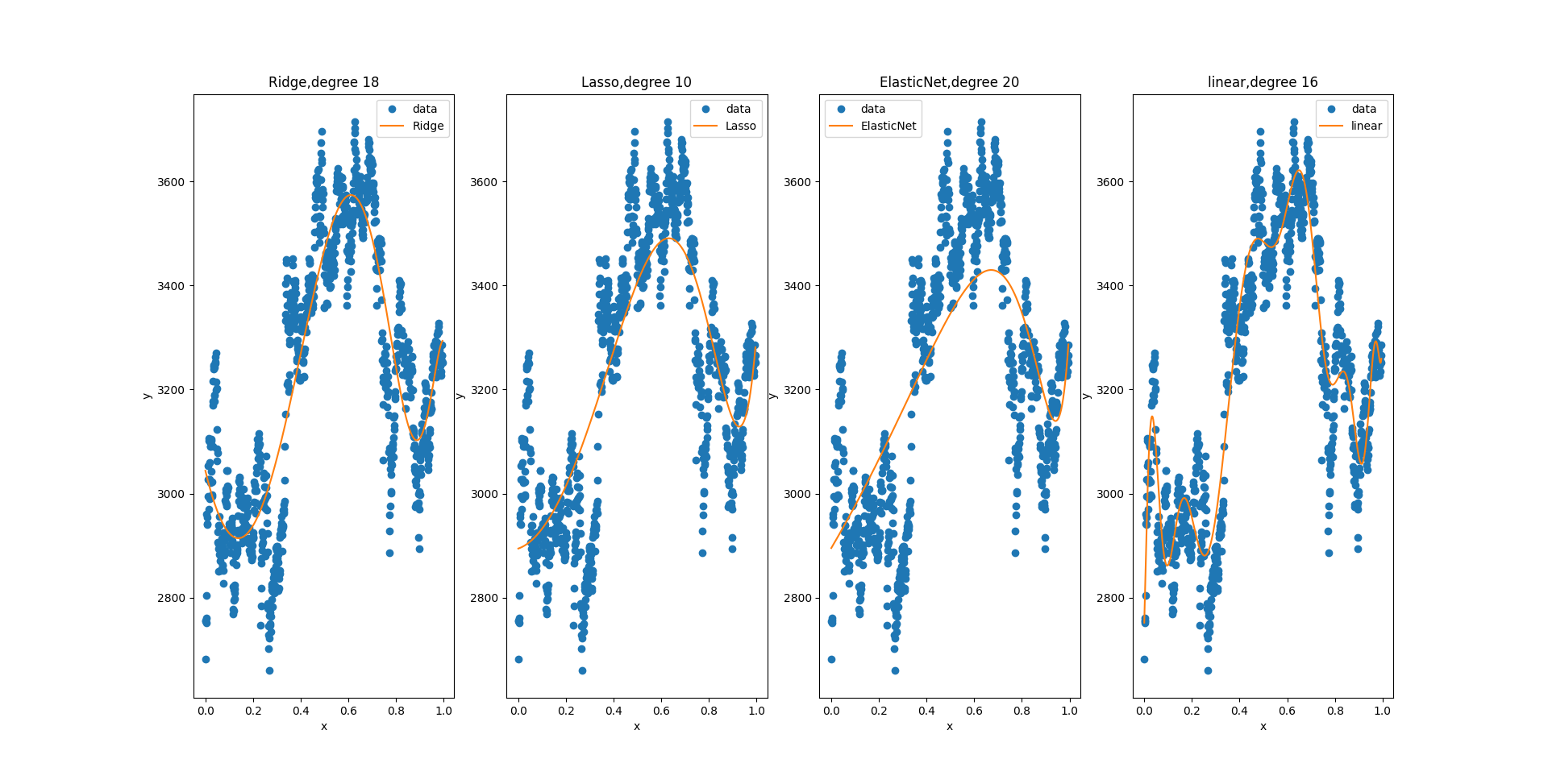
**lasso = Lasso(alpha=0.5, max\_iter=8000)**



而Ridge、elasticNet分别在18、20的参数上取得了较好的拟合结果：



最终拟合图像如下：



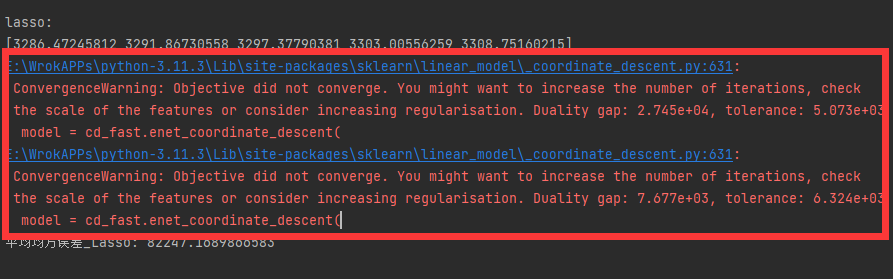
从图像中可以看到，在引入了损失函数后，即使多项式的参数变得很大，其图像相较于最右侧的基本线性回归，仍然相当平滑，可以推断模型明显具有了更强的泛化能力。

（四）

最后，通过交叉验证中的MSE指标来验证模型的分数。

由于cross\_val\_score函数所返回的是一个负值数组。因此这里采用平均取反的方式得到正均方误差作为评判标准，较低的均方误差代表着模型更高的预测准确度。

这里再次遇到了一个问题，尽管经过正则化参数和迭代次数的调整，Lasso模型实现了正常迭代，但是在10折交叉验证过程中，它再次出现了两次完全相同的未收敛报错：

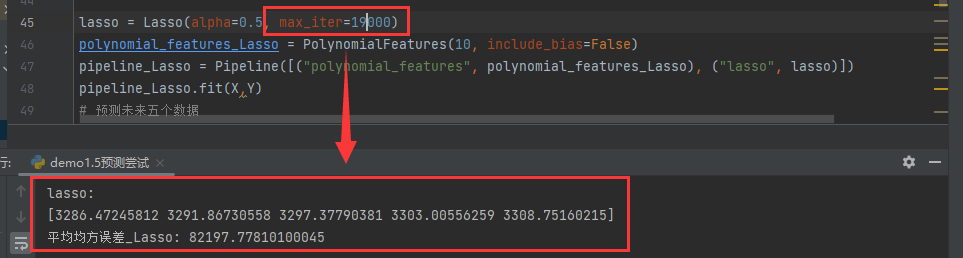


经过分析和查阅资料得知，这是由于交叉验证会将原数据集拆分为多个子集分别进行训练，由于其中某些子集的独特性质，导致我的模型在这些子集上收敛失败。根据两次报错可知，在10次训练中有两次子集训练是未收敛的。

尝试将Lasso模型的迭代次数由8000增加至10000后，可以看到，报错只剩下了一个。



将迭代次数逐渐增加至19000时，模型终于在所有子集训练上完成了收敛，同时有趣的是，MSE评分也有所降低，这代表模型的预测能力得到了些微的提升：



**六、代码附录**

（如有必要，此处可以粘贴完整的代码。）

【以下代码为普通线性回归代码】

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import cross\_val\_score

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import MinMaxScaler

# 这部分是demo1.1的导入数据

df = pd.read\_csv('C:/Users/24957/Desktop/指数.csv')

data = df["close"].values

data = data[6883:7883]

# 使用归一化准备X和Y的数据，这里选用了sklearn的归一化，y选用了后一千个数据

X = np.array([i for i in range(0, len(data) + 5)]).reshape(-1,1)

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

X\_pred = X[1000:]

X = X[:1000]

Y = data

degrees = [1, 5, 10, 17]

a = 0

for i in degrees:

# 老三样

ax = plt.subplot(1, len(degrees), a + 1) # 确认行列

a = a + 1

plt.setp(ax, xticks=(), yticks=()) # setp的好处是一次性可以设置多根线条的样式

linear\_regression = LinearRegression()

polynomial\_features = PolynomialFeatures(i, include\_bias=False)

pipeline = Pipeline([("polynomial\_features", polynomial\_features), ("linear\_regression", linear\_regression)])

pipeline.fit(X, Y)

# 通过交叉验证进行评分

scores = cross\_val\_score(pipeline, X, Y, cv=10)

# print(scores)

print(pipeline.predict(X\_pred))

# 画图

plt.plot(X, Y, 'o', label='data')

plt.plot(X, pipeline.predict(X), label="Model")

plt.xlabel("x")

plt.ylabel("y")

plt.legend(loc="best")

plt.title("Degree {}\n".format(i))

plt.show()

【以下代码为另外三种回归模型代码】

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression, Ridge ,Lasso ,ElasticNet

from sklearn.model\_selection import cross\_val\_score

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import MinMaxScaler

# 这部分是demo1.1的导入数据

df = pd.read\_csv('C:/Users/24957/Desktop/指数.csv')

data = df["close"].values

data = data[6883:7883]

# 使用归一化准备X和Y的数据，这里选用了sklearn的归一化，y选用了后一千个数据

X = np.array([i for i in range(0, len(data) + 5)]).reshape(-1,1)

scaler = MinMaxScaler()

ALL = scaler.fit\_transform(X)

X = ALL[:1000]

X\_predict = ALL[1000:]

Y = data

ridge = Ridge(alpha=0.01)

polynomial\_features\_Ridge = PolynomialFeatures(18, include\_bias=False)

pipeline\_Ridge = Pipeline([("polynomial\_features", polynomial\_features\_Ridge), ("ridge", ridge)])

pipeline\_Ridge.fit(X,Y)

# 预测未来五个数据

y\_pred = pipeline\_Ridge.predict(X\_predict)

print("\nridge:")

print(y\_pred)

ax = plt.subplot(1, 4, 1)

plt.plot(X, Y, 'o', label='data')

plt.plot(X, pipeline\_Ridge.predict(X), label="Ridge")

plt.xlabel("x")

plt.ylabel("y")

plt.title("Ridge,degree 18")

plt.legend(loc="best")

scores = cross\_val\_score(pipeline\_Ridge, X, Y, scoring="neg\_mean\_squared\_error", cv=10)

mean\_neg\_mse\_Ridge = np.mean(scores)

mean\_mse\_Ridge = -mean\_neg\_mse\_Ridge

print("平均均方误差\_Ridge:", mean\_mse\_Ridge)

lasso = Lasso(alpha=0.5, max\_iter=19000)

polynomial\_features\_Lasso = PolynomialFeatures(10, include\_bias=False)

pipeline\_Lasso = Pipeline([("polynomial\_features", polynomial\_features\_Lasso), ("lasso", lasso)])

pipeline\_Lasso.fit(X,Y)

# 预测未来五个数据

y\_pred = pipeline\_Lasso.predict(X\_predict)

print("\nlasso:")

print(y\_pred)

ax = plt.subplot(1, 4, 2)

plt.plot(X, Y, 'o', label='data')

plt.plot(X, pipeline\_Lasso.predict(X), label="Lasso")

plt.xlabel("x")

plt.ylabel("y")

plt.title("Lasso,degree 10")

plt.legend(loc="best")

scores = cross\_val\_score(pipeline\_Lasso, X, Y, scoring="neg\_mean\_squared\_error", cv=10)

mean\_neg\_mse\_Lasso= np.mean(scores)

mean\_mse\_Lasso = -mean\_neg\_mse\_Lasso

print("平均均方误差\_Lasso:", mean\_mse\_Lasso)

#

elasticNet = ElasticNet(alpha=0.01)

polynomial\_features\_ElasticNet = PolynomialFeatures(20, include\_bias=False)

pipeline\_ElasticNet = Pipeline([("polynomial\_features", polynomial\_features\_ElasticNet), ("elasticNet", elasticNet)])

pipeline\_ElasticNet.fit(X,Y)

# 预测未来五个数据

y\_pred = pipeline\_ElasticNet.predict(X\_predict)

print("\nelasticNet:" )

print(y\_pred)

ax = plt.subplot(1, 4, 3)

plt.plot(X, Y, 'o', label='data')

plt.plot(X, pipeline\_ElasticNet.predict(X), label="ElasticNet")

plt.xlabel("x")

plt.ylabel("y")

plt.title("ElasticNet,degree 20")

plt.legend(loc="best")

scores = cross\_val\_score(pipeline\_ElasticNet, X, Y, scoring="neg\_mean\_squared\_error", cv=10)

mean\_neg\_mse = np.mean(scores)

mean\_mse = -mean\_neg\_mse

print("平均均方误差\_ElasticNet:", mean\_mse)

linear\_regression = LinearRegression()

polynomial\_features = PolynomialFeatures(16, include\_bias=False)

pipeline = Pipeline([("polynomial\_features", polynomial\_features), ("linear\_regression", linear\_regression)])

pipeline.fit(X, Y)

print("\n")

print(pipeline.predict(ALL[1000:]))

ax = plt.subplot(1, 4, 4)

plt.plot(X, Y, 'o', label='data')

plt.plot(X, pipeline.predict(X), label="linear")

plt.xlabel("x")

plt.ylabel("y")

plt.title("linear,degree 16")

plt.legend(loc="best")

scores = cross\_val\_score(pipeline, X, Y, scoring="neg\_mean\_squared\_error", cv=10)

mean\_neg\_mse = np.mean(scores)

mean\_mse = -mean\_neg\_mse

print("平均均方误差\_Linear:", mean\_mse)

plt.show()