基于sklearn中线性回归模型机器学习方法对股票收盘曲线的拟合预测研究

随着居民财富和可投资资产的快速增长，如何让资产保值增值，是每个家庭和个人都会面临的问题。作为金融科技创新形式之一的智能量化投资，通过人工智能技术开展资产管理业务，能够大幅度提升资产管理的效率，正在成为中国金融业高质量发展的重要组成部分。而量化投资的策略基础是要对股票走势进行预测，本文将通过普通最小二乘法、岭回归、LASSO回归和弹性网络4中线性回归模型机器学习方法构建股票收盘价的预测模型，并对其进行的比较研究，并通过实际数据实例验证。

本文中模型的建立，先利用主成分分析对输入数据进行预处理得到降维后的数据，再利用处理后的数据通过各种回归模型机器学习方法计算得到拟合预测曲线。

**一、数据预处理**

主成分分析是一种统计方法。通过正交变换将一组可能存在相关性的变量转化为一组线性不相关的变量，转换后的这组变量称为主成分。

主成分分析主要是利用各个输入指数之间隐含的相关性，生成一个最接近原始数据的极大线性无关组，用较少的几个综合变量来代表原来的规模较大数据，从而做到了对数据的降维处理，减小了数据规模，能有效降低计算压力。具体处理步骤如下：

符号定义

|  |  |
| --- | --- |
| 符号 | 意义 |
|  | 原始数据矩阵 |
|  | 标准化矩阵 |
|  | 协方差矩阵 |
|  | 协方差矩阵的特征值 |
|  | 协方差矩阵的特征向量 |

1.通过公式(1)、(2)计算原始数据矩阵的平均值和标准差然后利用公式(3)对进行零均值标准化处理得到标准化矩阵。

2.利用公式(4)、(5)求标准化矩阵的协方差矩阵.。

3.根据公式(6)求出的n个特征值()，并按从大到小排序，，根据公式(7)求解相应特征值的单位特征向量。

4. 原始数据矩阵的主成分就是其协方差矩阵对应的特征向量，按照对应的特征值大小进行排序，最大的特征值就是第一主成分，其次是第二主成分，以此类推。按公式(8)计算累计方差贡献率，确定主成分个数k，累计方差贡献率≥95%的前k个主成分包含了原始数据中的绝大部分信息，这些数据作为机器学习的输入，后面的其他成分可以舍弃。

**二、模型设计**

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。具有很好的可解释性，可以从权重直接看出每个特征对结果的影响程度。线性回归模型的设计如下：

符号定义

|  |  |
| --- | --- |
| 符号 | 意义 |
|  | 实际值 |
|  | 输入数据 |
|  | 拟合值 |
|  | 拟合系数 |

其中

m为样本个数，n为变量个数

令

只需从线性方程组解出即可得到合适的拟合系数，从而进行对历史数据拟合和对未来数据预测。然而在实际中上述方程组一般情况下都是无解的，所以机器学习的过程是不断调整拟合系数，使得拟合值逐渐趋近实际值，线性回归模型的拟合方法主要有以下几种：

1.普通最小二乘法

普通最小二乘法是通过使数据集实际数据和预测之间的残差平方和最小来进行拟合，其最小化损失函数的数学表达式为

我们可以用以下公式直接求解

普通最小二乘法是一种无偏估计,通常是列满秩的时候，会获得比较好的拟合效果。但是当不是列满秩时，或者某些列之间的线性相关性比较大时，的行列式接近于0，接近于奇异，上述问题变成一个不适定问题，此时计算的误差会比较大，容易出现过拟合现象，所以普通最小二乘法缺乏稳定性和可靠性。

2.岭回归

为了克服普通最小二乘法缺乏稳定性和可靠性的缺点，我们需要将不适定问题转化为适定问题。因此需要对拟合系数施加惩罚，设惩罚系数为，岭回归在普通最小二乘法的损失函数的基础上增加了一个正则化项

岭回归通过使数据集实际数据和预测之间带惩罚项的残差平方和最小来进行拟合，其最小化损失函数的数学表达式为

其中。此时求解公式变为

上式中随着变化的轨迹称为岭迹，随着的增大，趋于0。模型中通常取W稳定时对应的值，即岭迹图中的稳定位置。

岭回归是一种有偏估计，虽然会使得残差的平方和变大，但是会使系数检验变好。岭回归是对普通最小二乘法的一种补充，当惩罚系数时退化为普通最小二乘法，它损失了无偏性，来换取高的数值稳定性，对病态数据的拟合要强于普通最小二乘法。

3. LASSO回归

LASSO (Least Absolute Shrinkage and Selection Operator)是一种以缩小变量集（降维）为主要思想的压缩估计方法。其通过构造一个惩罚函数得到一个较为精炼的模型，使得它压缩一些回归系数，即强制系数绝对值之和小于某个固定值；同时设定一些回归系数为零。与岭回归相似，LASSO 回归同样是通过添加正则项来改进普通最小二乘法，这里添加的正则化项为

其最小化损失函数的数学表达式为

这里可以使用坐标下降法来确定拟合系数。对于一个可微的凸函数，如果在某一点使得在每个坐标轴方向上都是最小的，那么就是一个全局最小值。于是优化目标就是在的个坐标轴上对损失函数做迭代下降，当所有坐标轴上对应都收敛时，损失函数最小。具体算法实现的过程如下：

(1)把向量随机取初始值，记为，代表第0轮迭代。

(2)对第k轮迭代，向量从开始依次求。

(3) 检查和向量在各个维度上的变化情况，如果所有维度的变化情况都比较小，结束迭代，否则继续k+1轮的迭代。

LASSO回归和岭回归都可以用来解决标准线性回归的过拟合问题。 二者的不同在于LASSO回归更容易使得权重变为0，而岭回归更容易使得权重接近0。岭回归是一个连续稳定的过程，但是该方法不能让大多数系数压缩到0，达到像LASSO回归那样减小因子的效果。从贝叶斯理论角度看，LASSO回归（使用L1 正则）等价于参数 w 的先验概率分布满足拉普拉斯分布，而岭回归（使用L2 正则）等价于参数 w 的先验概率分布满足高斯分布。

LASSO回归与普通最小二乘法相比虽然大大降低了预测方差，也达到了系数收缩和变量选择的目的，但是其还有一定的局限性。对于规模为的矩阵，其最多只能选出个变量，因此当即变量个数大于样本个数时，模型是为饱和的，处于一种过于稀疏的状态，得不到真实的结果。如果预测变量相互具有较强的依赖关系时，LASSO回归只能得到这组变量中的一个。如果预测变量中有很强的共线性时，LASSO回归的表现受控于岭回归。[4]

3.弹性网络回归

弹性网络是一种同时使用，范数作为先验正则项训练的线性回归模型。这种组合允许学习到一个只有少量参数是非零稀疏的模型，就像 LASSO回归一样，但是它仍然保持 一些像岭回归的正则性质。其最小化损失函数的数学表达式为

其中为惩罚系数。同样可以用和上面类似的坐标下降法实现。[5，6]

弹性网络的惩罚项为岭回归和LASSO回归惩罚项的凸组合。弹性网络在很多特征互相联系的情况下是非常有用的。LASSO回归很可能只随机考虑这些特征中的一个，而弹性网络更倾向于选择两个。在实践中，LASSO回归和岭回归之间权衡的一个优势是它允许在循环过程中继承岭回归的稳定性，既能做到岭回归不能做的特征提取，又能实现LASSO回归不能做的特征分组。

**三、误差分析指标**

本模型采用了如下6个常用的回归分析指标:

1. 解释方差分（Explained Variance Score）：这个指标用来衡量模型对数据集波动的解释程度，如果取值越接近1时，模型波动越小；如果取值越小，模型波动越大。计算公式如下:

2. 平均绝对误差（Mean Absolute Deviation, MSE）：又叫平均绝对离差，是所有单个观测值与算术平均值的偏差的绝对值的平均，平均绝对误差可以避免误差相互抵消的问题。计算公式如下：

3.均方误差（Mean Square Error, MSE）：即预测值与真实值之差的平方和的平均值，计算公式如下：

4. 均方对数误差（Mean Squared Logarithmic Error, MSLE）: 即预测值取对数与真实值取对数之差的平方和的平均值，计算公式如下：

5. 中位数绝对误差（Median Absolute Error, EedAE）：即所有预测值和真实值之差的绝对值的中位数。

6. 决定系数、R方：R方可以理解为因变量y中的变异性能能够被估计的多元回归方程解释的比例，它衡量各个自变量对因变量变动的解释程度，其取值在0与1之间，其值越接近1，则变量的解释程度就越高，其值越接近0，其解释程度就越弱。其计算公式如下：

**四、实证结果**

以股票代码为000001的平安银行为例，选取开盘、收盘、最低、最高、涨跌、涨幅、成交量、成交额/万元、MA5、MA10、MA20、sema、lema、diff、dea、macd共16个指标，按7:3切分数据训练集和验证集。

首先对数据进行预处理，对其股票的数据的16个指标进行主成分分析处理得到4个主成分，它们的累计贡献率分别为74.28、12.42%、6.86%、3.78%。累计贡献率达到了97.35%。分析结果如下：

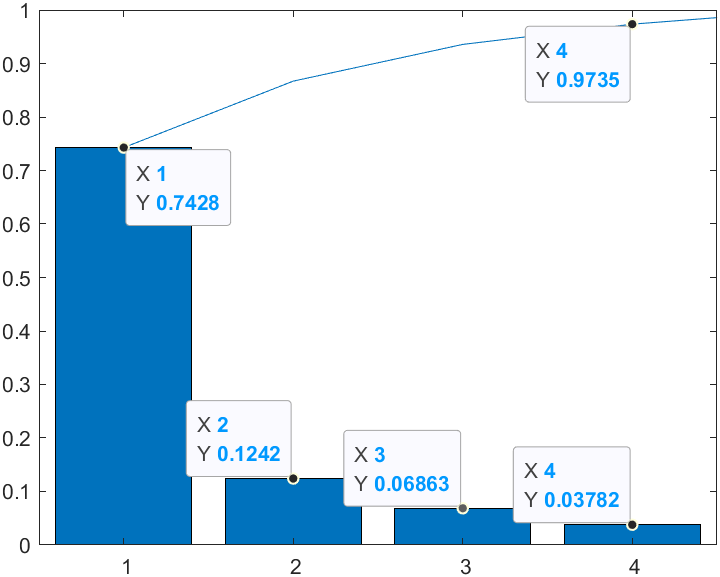


图1 各主成分累计贡献率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主成分 指标 | 第一主成分 | 第二主成分 | 第三主成分 | 第四主成分 |
| 开盘 | 0.2872 | -0.0145 | -0.0448 | -0.0939 |
| 收盘 | 0.2878 | 0.0390 | -0.0381 | -0.0643 |
| 最低 | 0.2874 | 0.0043 | -0.0413 | -0.0879 |
| 最高 | 0.2883 | 0.0137 | -0.0364 | -0.0551 |
| 涨跌 | 0.0444 | 0.6957 | -0.0671 | 0.0297 |
| 涨幅 | 0.0357 | 0.6981 | -0.0609 | 0.0552 |
| 成交量 | 0.2471 | -0.0578 | 0.1793 | 0.6017 |
| 成交额 | 0.2441 | -0.0460 | 0.2089 | 0.6106 |
| MA5 | 0.2880 | -0.0240 | -0.0694 | -0.0528 |
| MA10 | 0.2862 | -0.0258 | -0.1355 | -0.0354 |
| MA20 | 0.2826 | -0.0228 | -0.1941 | -0.0075 |
| sema | 0.2864 | -0.0206 | -0.1266 | -0.0334 |
| lema | 0.2837 | -0.0203 | -0.1659 | -0.0024 |
| diff | 0.2540 | -0.0194 | 0.2786 | -0.3252 |
| dea | 0.2549 | -0.0636 | 0.0162 | -0.3011 |
| macd | 0.0916 | 0.1191 | 0.8528 | -0.1894 |

图2 各主成分的指标组成

从上图中可以看出，在第一主成分中，除了涨跌、涨幅、macd之外占比较低之外，其他指标占比平均，故可以解释为稳定综合成分；第二主成分中涨跌、涨幅指标占比明显高于其他指标，故可以将其解释为价格波动成分；第三主成分中macd指标占比明显高于其他指标，故可以将其解释为移动平均成分；第三主成分中成交量、成交额指标占比明显高于其他指标，故可以将其解释为交易量成分。

以上这些主成分包含了原始数据中的绝大部分信息，将训练集作为机器学习的输入，在分别在普通最小二乘（OLS）、岭回归（Ridge）、LASSO回归和弹性网络（ElasticNat）模型训练，之后用验证集进行预测验证。黄线代表预测收盘价，蓝线代表实际收盘价，预测数据与实际数据的比对结果如下：

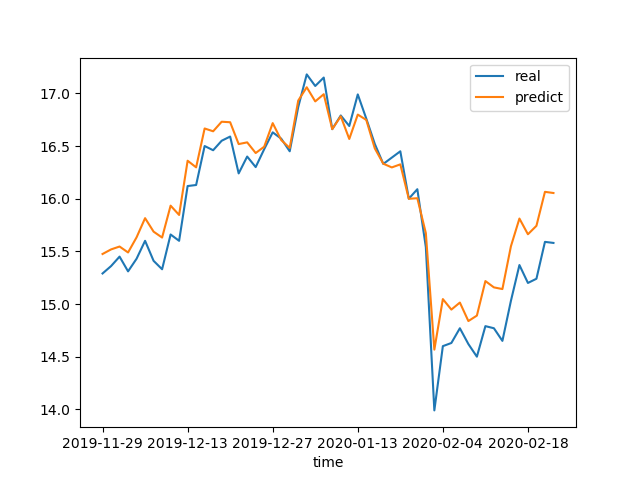


图3 普通最小二乘的预测结果

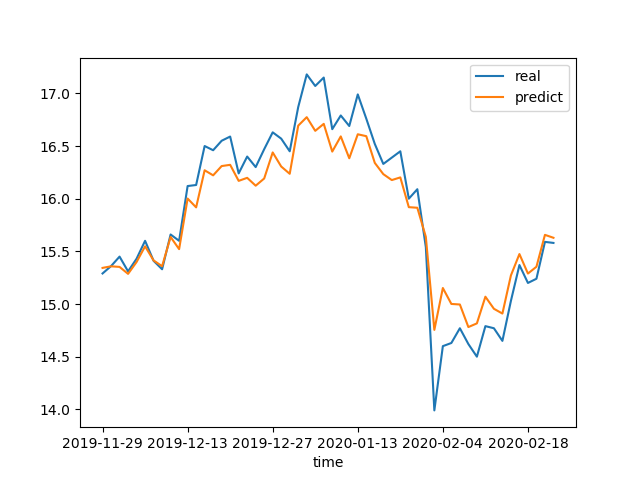


图4 岭回归的预测结果

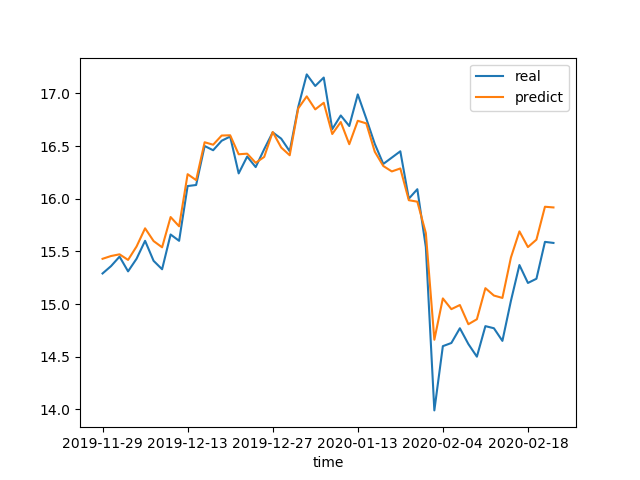


图5 LASSO的预测结果

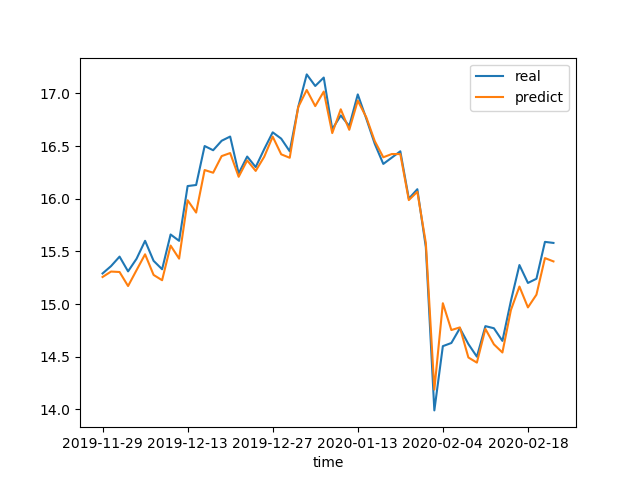


图6 弹性网络的预测结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 结果 | OLS | Ridge | LASSO | ElasticNet |
| 解释方差分 | 0.93643 | 0.91126 | 0.93772 | 0.98037 |
| 平均绝对误差 | 0.21456 | 0.19780 | 0.17359 | 0.10748 |
| 均方误差 | 0.07086 | 0.05973 | 0.05016 | 0.01775 |
| 均方对数误差 | 0.00027 | 0.00022 | 0.00019 | 0.00007 |
| 中位数绝对误差 | 0.17965 | 0.18786 | 0.13563 | 0.10664 |
| 决定系数 | 0.88990 | 0.90720 | 0.92206 | 0.97241 |

图7 预测结果误差分析

**五、结论分析与模型改进**

本文使用了sklearn库中4中线性回归模型机器学习方法，包括普通最小二乘法、岭回归、LASSO回归、弹性网络。并以原始数据经过主成分分析得到的数据成分作为输入，输出对股票收盘价格涨跌的预测。实证结果显示，4种线性回归模型的都获得了较好的预测效果，弹性网络模型的预测结果相比其他三种线性回归模型更好，有更高准确率和更小的误差。

由于数据来源等实验条件限制，模型任然有许多可以改进的地方，本文拟在如下三个方向深入研究并改进：

1.模型采用的数据来源于深圳证券官网，日线数据最早仅能获得到2019年5月21日，获取的股票历史数据较少。未来可以考虑将数据来源换成东方财富、wind数据等付费数据库获取更加完整的数据。

2.本模型的技术指标选择较少而且选择的数据大多比较普遍，缺乏完备性和完整性。比如缺少投资者投资情绪指标，未来模型完善可以通过文本学习增加投资者情绪。在技术指标的选择上可以通过一些更有效的特征选择方法筛选出蕴含更多股票信息、更能提升模型效果的技术指标。

3.本文仅研究了4中常见的线性回归模型机器学习方法，还有很多其他的机器学习方法如支持向量机、梯度提升树、集成神经网络、深度前馈网络、循环神经网络和长短期记忆网络等等未进行对比研究或结合运用，在面对更复杂的情形时可能无法获得很好的预测效果，未来可以通过研究多种方法综合运用，构建更好的模型以获得更加精确的预测效果。

参考文献

[1]周志华著.机器学习[M].北京：清华大学出版社.2016.

[2]（美）G.H.戈卢布（Gene H.Golub），（美）C.F.范洛恩（Charles F.Van Loan）著；袁亚湘等译.矩阵计算[M].北京：科学出版社.2001.

[3] 何宜盛. Lasso问题以及其在证券指数稀疏回归中的应用[D].南京大学,2016.

[4] “Regularization Path For Generalized linear Models by Coordinate Descent”, Friedman, Hastie & Tibshirani, J Stat Softw, 2010

[5] “An Interior-Point Method for Large-Scale L1-Regularized Least Squares,” S. J. Kim, K. Koh, M. Lustig, S. Boyd and D. Gorinevsky, in IEEE Journal of Selected Topics in Signal Processing, 2007

附录

完整代码链接：

预测模型主要代码：

import requests

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.preprocessing import scale

from sklearn.decomposition import PCA

from sklearn import linear\_model

from sklearn.linear\_model import ElasticNet

from sklearn.linear\_model import ElasticNetCV

from sklearn.metrics import explained\_variance\_score

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_squared\_log\_error

from sklearn.metrics import median\_absolute\_error

from sklearn.metrics import r2\_score

class Predicter(object):

    def \_\_init\_\_(self):

        # 从文件中读取原始数据

        self.fund\_code = "000001"

        self.origin\_data = pd.read\_csv(self.fund\_code+"日线数据.csv", encoding="gbk")

        # 绘图数据

        self.pic\_data = pd.DataFrame()

        self.pic\_data['time'] = self.origin\_data['时间']

        self.pic\_data['real'] = self.origin\_data['收盘']

        # 切分训练集和验证集

        self.origin\_data.drop(['时间'], axis=1, inplace=True)

        self.data\_process = self.pca\_process(self.origin\_data.values, 0.95)

        self.train\_data = np.array([])

        self.valid\_data = np.array([])

        self.train\_data = self.data\_process[:128]

        self.valid\_data = self.data\_process[128:]

        self.train\_target = self.origin\_data['收盘'].values[1:129]

    def pca\_process(self, data, proportion):

        if proportion > 1:

            return []

        pca = PCA()

        data = scale(data) # 数据标准化

        data = pca.fit\_transform(data)

        weight = 0

        i = 0

        while weight < proportion:

            weight += pca.explained\_variance\_ratio\_[i]

            i += 1

        # print(pca.explained\_variance\_ratio\_) # 主成分占比

        # print(pca.explained\_variance\_) # 特征根

        # print(np.linalg.eig(pca.get\_covariance())) # 协方差矩阵

        return data[:,0:i]

    def OLS(self):

        #  普通最小二乘

        model = linear\_model.LinearRegression()

        model.fit(self.train\_data, self.train\_target)

        # self.fit\_data = model.predict(self.train\_data)

        self.predict\_data = model.predict(self.valid\_data)

    def Ridge(self):

        # 岭回归

        model = linear\_model.Ridge(alpha=40)

        # model = linear\_model.RidgeCV(alphas=[0.1, 1.0, 10.0, 20.0, 40.0, 50.0])

        model.fit(self.train\_data, self.train\_target)

        # self.fit\_data = model.predict(self.train\_data)

        self.predict\_data = model.predict(self.valid\_data)

        # self.showModelAlpha(model)

    def LASSO(self):

        # LASSO回归

        model = linear\_model.Lasso(alpha=0.01)

        # model = linear\_model.LassoCV(alphas=[0.001, 0.005, 0.01, 0.1, 1.0, 10.0, 50.0])

        model.fit(self.train\_data, self.train\_target)

        self.fit\_data = model.predict(self.train\_data)

        self.predict\_data = model.predict(self.valid\_data)

        # self.showModelAlpha(model)

    def ElasticNet(self):

        # 弹性网络

        model = ElasticNet(alpha=0.03)

        # model = ElasticNetCV(alphas=[0.001, 0.005, 0.03, 0.1, 1.0, 10.0, 50.0])

        model.fit(self.train\_data, self.train\_target)

        # self.fit\_data = model.predict(self.train\_data)

        self.predict\_data = model.predict(self.valid\_data)

        # self.showModelAlpha(model)

    def showModelAlpha(self, model):

        print(model.alpha\_)

    def fit\_predictPic(self):

        # 预测和拟合曲线

        # self.pic\_data.drop(index=0, inplace=True)

        # self.predict\_data = self.predict\_data[:-1]

        # self.pic\_data.drop(index=len(self.pic\_data)-1, inplace=True)

        pic\_data = self.pic\_data

        pic\_data.set\_index(['time'],inplace=True)

        fit\_predict = np.concatenate((self.fit\_data, self.predict\_data))

        pic\_data['fit\_predict'] = fit\_predict

        pic\_data.plot()

        plt.show()

        return pic\_data

    def predictPic(self):

        # 预测曲线

        pic\_data = self.pic\_data

        # self.pic\_data.drop(index=len(self.pic\_data)-1, inplace=True)

        for i in range(len(self.train\_data)):

            pic\_data.drop(index=i, inplace=True)

        # self.predict\_data = self.predict\_data[:-1]

        print(len(self.predict\_data), len(pic\_data))

        pic\_data.set\_index(['time'],inplace=True)

        pic\_data['predict'] = self.predict\_data

        pic\_data.plot()

        plt.show()

        return pic\_data

if \_\_name\_\_ == "\_\_main\_\_":

    p = Predicter()

    # p.OLS()

    # p.Ridge()

    # p.LASSO()

    p.ElasticNet()

    # p.fit\_predictPic()

    # p.predictPic()

    # 误差分析

    data = p.predictPic()

    print(data)

    y\_true = data['real'].values

    y\_pred = data['predict'].values

    # 解释方差分

    EVS = explained\_variance\_score(y\_true, y\_pred)

    print("解释方差分:", EVS)

    # 平均绝对误差

    MAE = mean\_absolute\_error(y\_true, y\_pred)

    print("平均绝对误差:", MAE)

    # 均方误差

    MSE = mean\_squared\_error(y\_true, y\_pred)

    print("均方误差:", MSE)

    # 均方对数误差

    MSLE = mean\_squared\_log\_error(y\_true, y\_pred)

    print("均方对数误差:", MSLE)

    # 中位数绝对误差

    MAE\_ = median\_absolute\_error(y\_true, y\_pred)

    print("中位数绝对误差:", MAE\_)

    # 决定系数、R方

    r2 = r2\_score(y\_true, y\_pred)

    print("决定系数:", r2)