多种农产品价格智能预测

农产品价格影响农业生产的发展，农产品的流通、消费和农民的收入水平，而且影响工业品的成本和价格，影响国家同农民之间、城乡人民之间以及农民内部的物质利益关系，对整个社会经济生活的安定也关系重大。农产品价格受到气候、供求关系、宏观经济等因素的影响后波动性较大。有关国内农产品涨价会影响大众生活质量。如果能预测某些农产品的价格，并针对某些波动幅度大的农产品价格进行宏观调控与管理，将在一定程度上解决广大农民的生计问题。本项目数据来源于2016 年以前全国各农场品交易市场。目的是利用填补后的数据构建模型，对所选取出的 935 种农作物价格做出预测。

本项目主要涉及Min-Max标准化、ANN（人工神经网络）等方法。

**学习目标**

1. 练习利用Pandas 库进行数据预处理
2. 练习缺少值填充。
3. 利用sklearn模块建模模型并预测

# 了解农产品数据与分析流程

**任务描述**

通过对历史数据的分析，预测未来的农产品价格走势，为决策者提供更有力的数据支持，从而制定出更为科学合理的政策。

**任务分析**

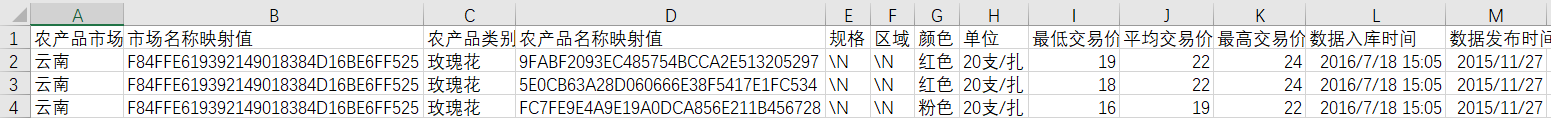
可以通过以下4个步骤了解所给农产品数据与分析流程。

1. 了解所给数据，分析数据直接的关联
2. 选取要用于预测的数据
3. 熟悉数据分析的步骤与方法

## 了解所给的数据

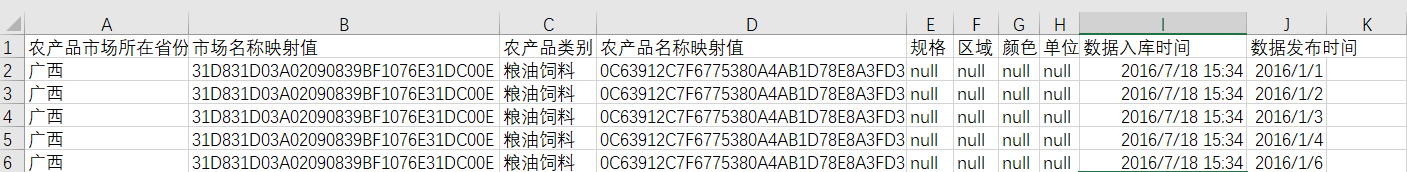
Farming2.csv文件的列包含了农产品市场所在省份、市场名称映射值、农产品类别、农产品名称映射值、规格、区域、颜色、单位、最低交易价格、平均交易价格、最高交易价格、数据入库时间、数据发布时间。市场名称映射值唯一确定市场名称。存在市场名称映射值相同、类别相同、颜色、单价相同，但农产品映射值不同的数据。项目发布时间均在2016年之前。某些数据入库时间在2016年。在有些行存在平均交易价格大于最高价的现象。最低交易价格、平均交易价格、最高交易价格。

图 1‑1 excel打开的farming2.csv部分示例



在product\_market.csv文件中给出了需要预测的时间、市场名称映射值、农产品名称映射值。这两种映射值可以与之前的数据对应。预测的时间大致集中在一个月的时间之内。表中没有给出区域和规格信息，这对后续数据分析没有影响。

图 1‑2 用excel打开的product\_market.csv部分示例



## 预测时需要的数据

市场名称映射值唯一确定市场名称。农产品名称预测值可以对应确定单价和颜色。所以不考虑单价、颜色。数据入库时间也对后续分析没有影响。

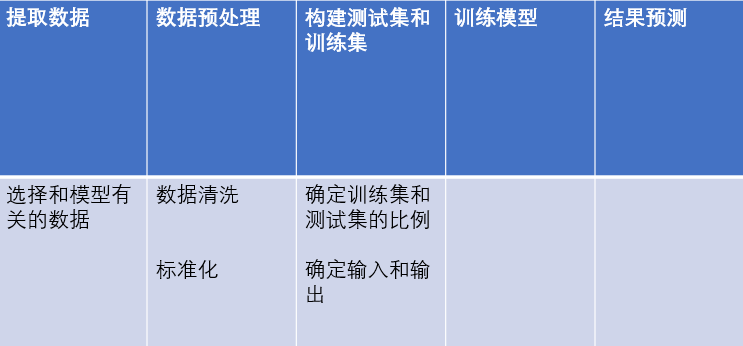
后续主要使用的是表中的映射值和数据发布时间、最低交易价格、平均交易价格、最高交易价格等指标。

## 熟悉数据分析的步骤与流程

农产品数据分析总体流程如下所示，主要包括以下4个步骤。

1. 提取建立模型需要的数据
2. 对抽取的数据进行预处理，进行标准化、数据清洗（去除某些无关列，如规格）等操作。
3. 利用sklearn构成训练集和测试集
4. 构建的ANN（Artificial Neural Network）进行预测
5. 整理预测结果，让结果更直观

图 1‑3 农产品价格预测大致流程



# 预处理农产品数据

**任务描述**

原始数据存在异常值、无关值、不标准类型。异常值包含了同一时间同一农产品不同价格、最高交易价与最低交易价均为0而平均交易价不为0的数据；无关行比如规格、颜色等；无关值指与需预测农产品无关的数据；不标准类型指时间。运用预处理能够减少数据处理量、避免异常数据对预测模型的干扰、从而有助于提高其后的模型预测的准确率和效率，对于本项目十分必要。

**任务分析**

对预处理农产品数据可分为以下步骤。

1. 数据录入
2. 数据去重
3. 处理无关行
4. 处理无关农产品
5. 处理异常值
6. 处理矛盾值
7. 处理时间序列

## 数据录入

首先使用pd.read\_csv函数录入farming2.csv文件及需预测的product\_market.csv文件。

## 数据去重

为避免大量重复信息提高数据处理量，本项目对信息相同的数据进行去重处理。farming2.csv文件原有信息2103048行，经去重后剩余1951933行。去重结果合理，减少了数据处理量。

## 处理无关行

通过研究数据，发现farming2.csv文件中的农产品市场所在省份 、规格、区域、颜色、单位、数据入库时间信息对预测结果无影响，为了减少数据处理量，加快代码运行速度，本项目选择用drop函数去除无关行。



图2-1删除无关行结果

## 处理无关农产品

本项目中farming2.csv文件中，数据远远多于product\_market.csv文件，除考虑数据提供方提供了较多数据供预测，提高结果的科学性、有效性外，farming2.csv文件中存在大量无需预测的无关农产品也可能大大提高farming2.csv文件的数据量。

通过查找农产品映射值及市场名称映射值，发现farming2.csv文件中存在6410种农产品映射值和市场名称映射值组合，而product\_market.csv文件中仅有1337种组合。由此断定farming2.csv文件中存在无关农产品。

除此之外，由于不能确定需要预测的农产品farming2.csv文件中是否都提供了，本项目通过查找两个文件中农产品映射值及市场名称映射值组合的相同部分，得到两文件相同的农产品映射值及市场名称映射值组合共1337种，从而判定farming2.csv文件含有全部需预测的农产品信息。

为避免无关数据提高数据处理量，本项目将相关的农产品映射值提取为data\_code.csv文件，便于第3小节的预测。



图2-2部分有关农产品映射值

## 处理异常值

farming2.csv文件中部分农产品交易价格带有单位，会干扰本项目的数据处理。通过观察发现，仅存在带有（个）单位的数据。为减少代码工作量，寻求更为简单直接的解决方式，本项目通过excel带有的替换功能，将单位（个）替换为空，从而删除了单位。

通过观察，发现在给定数据中，在farming2.csv文件中有许多数据的平均交易价格为0。最高交易价、最低交易价、平均交易价格都为0的数据，表明农产品在当日未进行交易，该类数据不属于异常值。若最高交易价和最低交易价都不为0，平均交易价为0，则表明出现异常值。对于该类异常数据，本项目使用最高交易价格和最低交易价格的平均值来填充平均交易价格。

图2-3平均值填补部分结果

本项目筛选出最高交易价和最低交易价都不为0，平均交易价为0的数据并填补了平均交易价格后，将填补后的数据与原数据进行合并，删除未填补的数据，并测试是否填充成功。填充后筛选最高交易价和最低交易价都不为0，平均交易价为0的数据，得到0行数据，说明填充成功。

## 处理矛盾值

如果存在市场名称映射值、农产品名称映射值、数据发布时间均相同，但交易价格不同的数据，即存在同一时间同一农产品的不同价格，则表明数据收集或录入过程中出现了差错，对于这类矛盾值，本项目依据市场名称映射值、农产品名称映射值、数据发布时间进行分组，将每组交易价格取平均后仅保留平均值。



图2-4矛盾值处理部分结果

## 处理时间序列

运用pd.to\_datetime()，可以将对应的列转换为Pandas中的datetime64类型，便于后期的处理。

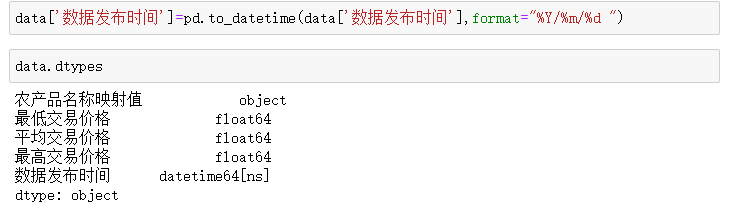


图2-5时间标准化结果

# 使用ANN进行农产品价格预测

**任务描述**

对数据预处理完成之后，为了对农产品进行价格预测需要根据所给的数据进行模型的拟合。本小节主要分析模型的选择与构建，并对所构建的模型进行调参与评价。构建好模型后对所选取的935种农作物价格做出预测。

**任务分析**

1. 分析农产品价格与时间的变化及其内在联系，选择合适的模型进行拟合
2. 对所选择的模型进行构建
3. 对模型进行调参并对构建好的模型进行评估
4. 模型应用

## 模型的选择

在模型的选择上可以考虑线性回归模型、多项式拟合、长短期记忆网络、人工神经网络。

⑴线性回归模型是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。通过对数据的观察与分析，农产品的时间与价格的线性关系并不强，故舍弃此方法。

⑵多项式拟合是用一个多项式展开去拟合包含数个分析格点的一小块分析区域中的所有观测点，得到观测数据的客观分析场，展开系数用最小二乘拟合确定。通过对随机抽取的一个样本进行七次多项式拟合如图 3‑1（蓝色线条为实际价格，橙色线条为预测价）。可以发现，拟合的误差率较大，且会在之后的预测中价格会呈持续下降趋势，甚至出现负数，故舍弃此法。

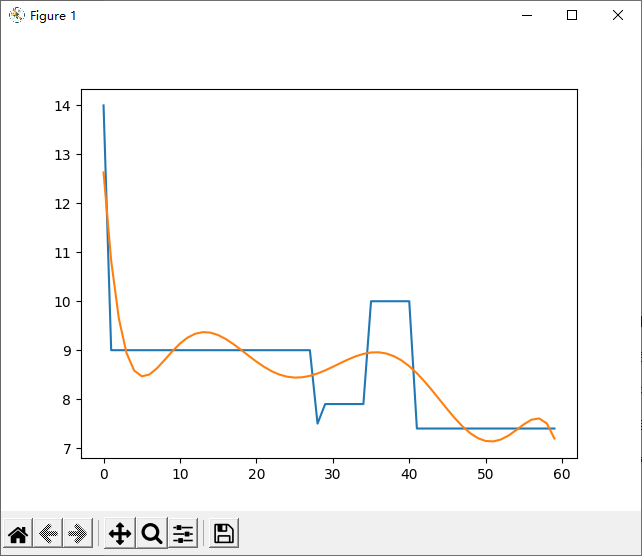


图 3‑1 多项式拟合

⑶长短期记忆网络是一种递归神经网络（RNN），通过训练时在“时间上的反向传播”来克服梯度消失问题，可以用来构建大规模的递归神经网络来处理机器学习中复杂的序列问题，并取得不错的结果。但是长短期记忆网络对时间序列要求比较苛刻，需要时间序列具有稳定性，即保证时间序列的统计特征如平均数，方差随着时间保持不变。本项目大多数时间序列并不满足此条件，且不同产品的数据样本容量也相差较大，短则几个月，长则几年，不适合使用长短期记忆网络，故舍弃此法。

⑷人工神经网络可以根据环境的变化，对权值进行调整，改善系统的行为，通过其自身的自学习功能去不断地逼近每个产品时间与价格的关系函数，在得到产品一个月中价格的变化趋势后，即可对下一个月的价格进行预测。且人工神经网络对网络的输入只要求介于0至1之间，可以通过对时间的标准化达到此要求。故选择此种模型。

## 模型构建

数据的读取及处理,清洗后的样本处于data\_gp.csv文件中，使用pandas进行csv文件的读取,将时间转换为timestamp类型，并将日期赋给x，将平均交易价格赋给y。

代码 3‑1 数据的读取及预处理

|  |
| --- |
| import numpy as np  import pandas as pd  data2 = pd.read\_csv('data\_gp.csv') # 导入data\_gp.csv文件  data3 = data2.loc[data2['农产品名称映射值'] == name[k]] # 从data\_gp.csv中找到需要预测的农产品的训练数据  data3 = data3.loc[data3['市场名称映射值'] == loc[k]]  data3 = pd.DataFrame.drop\_duplicates(data3,subset='数据发布时间',keep='last') # 按时间去重  data3 = data3.sort\_values(by = '数据发布时间',ascending=True) # 按日期排序  data3['数据发布时间'] = pd.to\_datetime(data3['数据发布时间']) # 时间处理  x = data3['数据发布时间'].dt.day # 提取发布时间的日期  y = data3['平均交易价格'] # 提取平均交易价格 |

ANN要求数据的输入介于0-1之间，首先对时间进行离差标准化。通过sklearn库中的minmax\_scale函数实现。为了方便后续模型的数值处理，将其转换为numpy数组。

代码 3‑2 时间的离差标准化

|  |
| --- |
| from sklearn import preprocessing  x1 = preprocessing.minmax\_scale(x)  y1 = preprocessing.minmax\_scale(y)  x1 = np.array(x1) # 将时间列转化为numpy数组  y1 = np.array(y1) # 将平均交易列转化为numpy数组 |

进行模型的构建，激活函数选择sigmoid函数，隐层神经元为四个，阈值初始值为-1，一个日期输入对应一个价格输出，所以输入输出值个数都为1，建立一个三层神经网络。

代码 3‑3 模型的构建

|  |
| --- |
| # 模型训练  def sigmoid(x): # 网络激活函数  return 1/(1+np.exp(-x))  yita = 0.15 # 学习速率  out\_in = np.array([0.0,0,0,0,-1]) #输出层的输入  w\_mid = np.zeros([2,4]) #隐层神经元的权值&阈值  w\_out = np.zeros([5]) # 输出层神经元的权值&阈值  delta\_w\_out = np.zeros([5])  delta\_w\_mid = np.zeros([2,4])  Err = []  for j in range(1000):  error = []  for it in range(x\_train.size):  net\_in = np.append(x\_train[it], -1) # 网络输入  real = y\_train[it]  for i in range(4):  out\_in[i] = sigmoid(sum(net\_in \* w\_mid[:, i])) #从输入到隐层的传输过程  res = sigmoid(sum(out\_in \* w\_out)) # 模型预测值  error.append(abs(real - res))  delta\_w\_out = yita\*res\*(1-res)\*(real-res)\*out\_in # 输出层权值的修正量  delta\_w\_out[4] = - yita\*res\*(1-res)\*(real-res) # 输出层阈值的修正量  w\_out = w\_out + delta\_w\_out # 更新  for i in range(4):  delta\_w\_mid[:, i] = yita\*out\_in[i]\*(1-out\_in[i])\*w\_out[i]\*res\*(1-res)\*(real-res)\*net\_in # 中间层神经元的权值修正量  delta\_w\_mid[1,i] = -yita\*out\_in[i]\*(1-out\_in[i])\*w\_out[i]\*res\*(1-res)\*(real-res) #中间层神经元的阈值修正量  w\_mid = w\_mid + delta\_w\_mid # 更新  Err.append(np.mean(error)) # 计算平均误差 |

## 模型调参及评估

对模型进行调参即对模型中的学习速率yita以及训练的轮数进行调整，使其与真实值的平均误差尽可能的减小。

代码 3‑4 计算平均误差

|  |
| --- |
| Err.append(np.mean(error)) # 计算平均误差 |

对模型进行评估需要从清洗后的数据中划分训练集和测试集，然后将测试集带入训练集中计算平均误差，以此来评估模型的拟合效果。

代码 3‑5 划分训练集和测试集

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  # 划分训练集和测试集  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x1,y1,test\_size=0.2) |

代码 3‑6 模型评估

|  |
| --- |
| # 代入测试集进行模型评估  error\_te = []  for it in range(x\_test.size):  net\_in = np.append(x\_test[it], -1) # 网络输入  real = y\_test[it]  for i in range(4):  out\_in[i] = sigmoid(sum(net\_in \* w\_mid[:, i])) # 从输入到隐层的传输过程  res = sigmoid(sum(out\_in \* w\_out)) # 模型预测值  error\_te.append(abs(real - res))  print('模型测试集平均误差为：',np.mean(error\_te)) |

经过参数调整后可以看到，模型在训练集中的平均误差率随训练的轮数逐渐减小。放到测试集中平均误差率约在0.25左右。

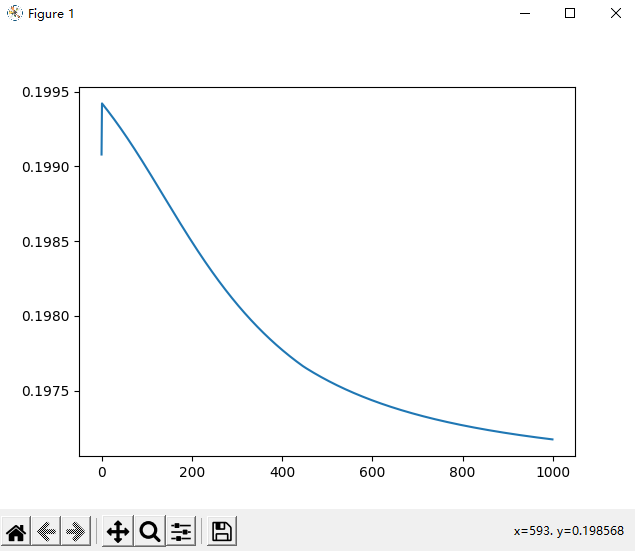


图 3‑2 训练集平均误差率

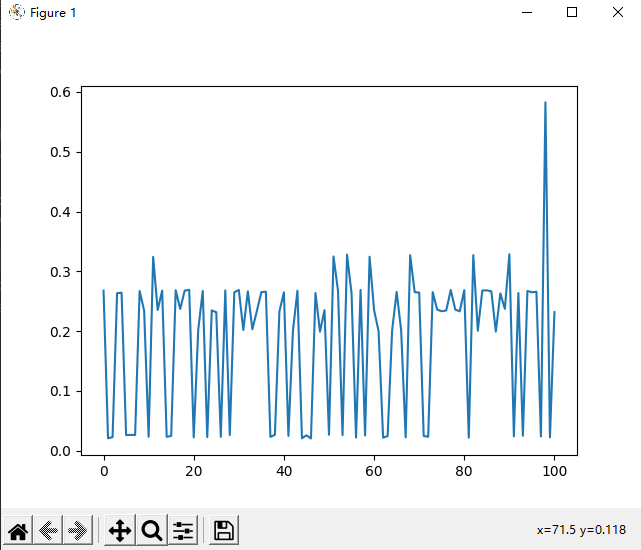


图 3‑3 实际样本平均误差率

## 模型应用

通过for循环将需要预测的农产品的产品名称和市场名称映射值依次导入python中在所给的数据集里进行匹配，找到训练数据后导入模型进行训练，然后对一个月中每天的价格进行预测并将预测价格进行输出，结果保存于excel文件中

代码 3‑7 模型应用

|  |
| --- |
| x\_true = np.array(range(32))  x\_true = x\_true[1:32]  x\_true = preprocessing.minmax\_scale(x\_true)  for it in range(31):  net\_in = np.append(x\_true[it], -1) # 网络输入  for i in range(4):  out\_in[i] = sigmoid(sum(net\_in \* w\_mid[:, i])) #从输入到隐层的传输过程  res = sigmoid(sum(out\_in \* w\_out)) # 模型预测值  y\_pre = res \* (y.max()-y.min())+y.min() #标准化还原  print(name[k],'产品在',it+1,'号的预测价格为',y\_pre) |

# 小结

本次项目旨在通过数据分析及建模对农产品价格进行预测。通过运用numpy、pands库对数据进行预处理，缺少值填充，利用sklearn进行模块建模并预测。对python的运用，数据处理的方法，模型的建立，机器学习等都有了一定程度的理解及掌握。重要的是对整个数据分析与应用的流程有了较为深刻的认识。

**项目完整代码：**

**数据预处理部分**

|  |
| --- |
| **import** pandas **as** pd **import** numpy **as** np **from** pandas **import** to\_datetime **import** matplotlib.pyplot **as** plt **from** scipy.interpolate **import** interp1d *# 读取farming2.csv及product\_market.csv文件* data=pd.read\_csv(**'C:\\Users\\LYF\\Desktop\\farming2.csv'**,encoding=**'gbk'**) data2=pd.read\_csv(**'C:\\Users\\LYF\\Desktop\\product\_market.csv'**) *# 查看数据长度* data.shape *# 去除重复行* data.drop\_duplicates(subset=**None**, keep=**'first'**, inplace=**True**) *# 查看去重后数据长度* data.shape *# 删除无关行* data.drop(data.columns[[0,2,4,5,6,7,11]],axis=1,inplace=**True**) *# 查看farming2.csv文件农产品映射值和市场名称映射值组合种类* data\_code=data.drop\_duplicates(subset=data.columns[[0,1]], keep=**'first'**, inplace=**False**) data\_code.shape *# 查看product\_market.csv文件农产品映射值和市场名称映射值组合种类* data2\_code=data2.drop\_duplicates(subset=[**'农产品名称映射值'**,**'市场名称映射值'**], keep=**'first'**, inplace=**False**) data2\_code.shape *#查找并导出相同农产品映射值  #合并2个表中的组合种类* datacd\_sum=pd.concat([data\_code,data2\_code])  *#保留了第一次出现的相同项和不同项* datacd\_same=datacd\_sum.drop\_duplicates(subset=[**'农产品名称映射值'**,**'市场名称映射值'**],keep=**'first'**, inplace=**False**)  datacd\_same.shape *#保留了不同项* datacd\_norepeat=datacd\_sum.drop\_duplicates(subset=[**'农产品名称映射值'**,**'市场名称映射值'**],keep=**False**, inplace=**False**)  datacd\_norepeat.shape *#相同＋2\*不同* datacd\_sum2=pd.concat([datacd\_same,datacd\_norepeat]) *#相同＋2\*不同-2\*不同* data\_cd=datacd\_sum2.drop\_duplicates(subset=[**'农产品名称映射值'**,**'市场名称映射值'**],keep=**False**, inplace=**False**) data\_cd.to\_csv(**'C:\\Users\\LYF\\Desktop\\data\_code.csv'**) *#对多个同一时间的同一商品价格取平均，以平均值代替多组数据* data\_copy=data data\_copy.groupby([**'市场名称映射值'**,**'农产品名称映射值'**,**'数据发布时间'**])[[**'平均交易价格'**,**'最高交易价格'**,**'最低交易价格'**]].mean() data\_group=data\_copy.groupby([**'市场名称映射值'**,**'农产品名称映射值'**,**'数据发布时间'**])[[**'平均交易价格'**,**'最高交易价格'**,**'最低交易价格'**]].mean() *#平铺数据* data\_group=data\_group.reset\_index()  *#测试是否替换成功* data.drop\_duplicates(subset=[**'市场名称映射值'**,**'数据发布时间'**,**'农产品名称映射值'**],keep=**'first'**,inplace=**True**) data.shape *#若平均值为0，最高最低值不为0，将最高最低值取平均后填补平均值 #选取数据取平均* data\_pj=data\_group.loc[(data\_group[**'平均交易价格'**] == 0)& (data\_group[**'最高交易价格'**]!= 0)&(data\_group[**'最低交易价格'**]!= 0)] data\_pj[**'平均交易价格'**]=(data\_pj[**'最高交易价格'**]+data\_pj[**'最低交易价格'**])/2 data\_pj *#与原数据合并，删除未填补的数据* data\_gpsum=pd.concat([data\_pj,data\_group]) data\_gp=data\_gpsum.drop\_duplicates(subset=[**'市场名称映射值'**,**'数据发布时间'**,**'农产品名称映射值'**],keep=**'first'**) data\_gp *#测试是否成功填充* data\_pjtext=data\_gp.loc[(data\_gp[**'平均交易价格'**] == 0)& (data\_gp[**'最高交易价格'**]!= 0)&(data\_gp[**'最低交易价格'**]!= 0)] data\_pjtext.shape *#时间标准化* data[**'数据发布时间'**]=pd.to\_datetime(data[**'数据发布时间'**],format=**"%Y/%m/%d "**) *#检测是否标准化成功* |

**建模部分**

|  |
| --- |
| **import** numpy **as** np **import** pandas **as** pd **import** matplotlib.pyplot **as** plt **from** sklearn **import** preprocessing **from** sklearn.model\_selection **import** train\_test\_split   pd.set\_option(**'display.max\_columns'**,1000) pd.set\_option(**'display.width'**,1000)  data1 = pd.read\_csv(**'data\_code.csv'**,encoding=**'gbk'**) *# 导入data\_code.csv文件* data2 = pd.read\_csv(**'data\_gp.csv'**) *# 导入data\_gp.csv文件* name = data1[**'农产品名称映射值'**] loc = data1[**'市场名称映射值'**] name = name.to\_numpy() loc = loc.to\_numpy()  **for** k **in** range(name.size):  data3 = data2.loc[data2[**'农产品名称映射值'**] == name[k]] *# 从data\_gp.csv中找到需要预测的农产品的训练数据* data3 = data3.loc[data3[**'市场名称映射值'**] == loc[k]]  data3 = pd.DataFrame.drop\_duplicates(data3,subset=**'数据发布时间'**,keep=**'last'**) *# 按时间去重* data3 = data3.sort\_values(by = **'数据发布时间'**,ascending=**True**) *# 按日期排序* data3[**'数据发布时间'**] = pd.to\_datetime(data3[**'数据发布时间'**]) *# 时间标准化* x = data3[**'数据发布时间'**].dt.day *# 提取发布时间的日期* y = data3[**'平均交易价格'**]   **def** unique\_cols(df): *# 判断平均价格是否一直保持不变* a = df.to\_numpy()  **return**(a[0] == a[1:]).all(0)   **if** unique\_cols(y) == 1: *# 如果是，则预测价格也保持不变* y = y.to\_numpy()  y\_pre = y[0]  print(name[k],**'产品预测价格为:'**,y\_pre)  print(**'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'**)   **else**: *# 不是则进行价格预测* x = x.astype(float)  y = y.astype(float)   x1 = preprocessing.minmax\_scale(x)  y1 = preprocessing.minmax\_scale(y)   x1 = np.array(x1) *# 将时间列转化为一维数组* y1 = np.array(y1) *# 平均交易价格设为y，并将该列转化为一维数组    # 划分训练集和测试集* x\_train,x\_test,y\_train,y\_test = train\_test\_split(x1,y1,test\_size=0.2)   *# 模型训练* **def** sigmoid(x): *# 网络激活函数* **return** 1/(1+np.exp(-x))   yita = 0.15 *# 学习速率* out\_in = np.array([0.0,0,0,0,-1]) *#输出层的输入* w\_mid = np.zeros([2,4]) *#隐层神经元的权值&阈值* w\_out = np.zeros([5]) *# 输出层神经元的权值&阈值* delta\_w\_out = np.zeros([5])  delta\_w\_mid = np.zeros([2,4])  Err = []  **for** j **in** range(1000):  error = []  **for** it **in** range(x\_train.size):  net\_in = np.append(x\_train[it], -1) *# 网络输入* real = y\_train[it]   **for** i **in** range(4):  out\_in[i] = sigmoid(sum(net\_in \* w\_mid[:, i])) *#从输入到隐层的传输过程* res = sigmoid(sum(out\_in \* w\_out)) *# 模型预测值* error.append(abs(real - res))    delta\_w\_out = yita\*res\*(1-res)\*(real-res)\*out\_in *# 输出层权值的修正量* delta\_w\_out[4] = - yita\*res\*(1-res)\*(real-res) *# 输出层阈值的修正量* w\_out = w\_out + delta\_w\_out *# 更新* **for** i **in** range(4):  delta\_w\_mid[:, i] = yita\*out\_in[i]\*(1-out\_in[i])\*w\_out[i]\*res\*(1-res)\*(real-res)\*net\_in *# 中间层神经元的权值修正量* delta\_w\_mid[1,i] = -yita\*out\_in[i]\*(1-out\_in[i])\*w\_out[i]\*res\*(1-res)\*(real-res) *#中间层神经元的阈值修正量* w\_mid = w\_mid + delta\_w\_mid *# 更新* Err.append(np.mean(error)) *# 计算平均误差*   *# 代入测试集进行模型评估* error\_te = []  **for** it **in** range(x\_test.size):  net\_in = np.append(x\_test[it], -1) *# 网络输入* real = y\_test[it]   **for** i **in** range(4):  out\_in[i] = sigmoid(sum(net\_in \* w\_mid[:, i])) *# 从输入到隐层的传输过程* res = sigmoid(sum(out\_in \* w\_out)) *# 模型预测值* error\_te.append(abs(real - res))   print(**'模型测试集平均误差为：'**,np.mean(error\_te))   *# 价格预测* x\_true = np.array(range(32))  x\_true = x\_true[1:32]  x\_true = preprocessing.minmax\_scale(x\_true)  **for** it **in** range(31):  net\_in = np.append(x\_true[it], -1) *# 网络输入* **for** i **in** range(4):  out\_in[i] = sigmoid(sum(net\_in \* w\_mid[:, i])) *#从输入到隐层的传输过程* res = sigmoid(sum(out\_in \* w\_out)) *# 模型预测值* y\_pre = res \* (y.max()-y.min())+y.min() *#标准化还原* print(name[k],**'产品在'**,it+1,**'号的预测价格为'**,y\_pre)   print(**'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'**) |