2018年江西省研究生数学建模竞赛

参赛队号： 2018266B

题目： 农产品滞销风险评估问题

农产品滞销风险评估问题

摘要

目前农产品滞销受到社会各界以及政府的广大关注，然而目前已有的方法只可解一时之急，治标而不治本。因此，本文通过筛选出影响农产品滞销的几个主要因素作为主要评价指标，构建较为合理的农产品滞销风险评估模型来预测农产品的价格，并计算其在一定时间内价格的波动程度，接着将农产品价格波动幅度进行标准化处理作为农产品滞销风险的量化结果并根据结果分布情况将风险划分为“很大、大、不大、小、很小”5个等级，最后结合市场应用背景，向农业管理部门提供了保证该预测方法有效实施的建议。

对于问题一，通过“新浪舆情”提供的《农产品滞销数据分析报告》、“中国供销合作社”发布的近一年的农产品市场动态信息和42个大型农产品批发市场价格进行统计分析及自然语言处理（NLP）的词频分析，得出关于影响农产品滞销的几个最主要因素：农产品上市期间的天气、市场行情（包括市场价格、供求关系）、城市交通密度（体现农作物运输的难易程度）、地区、农产品的类型。

对于问题二，对主要滞销因素进行相关性及可视化分析，利用one-hot及归一化方法将收集到的数据数值化，在使用多个机器学习后，最后选用决策树回归及时间序列分析融合的方式建立模型，得出预测农产品的价格的数学模型并能计算其在一定时间内波动程度，接着通过对农产品价格波动幅度进行标准化将风险划分为“很大、大、不大、小、很小” 评价农产品滞销风险的5个等级。

对于问题三，我们对所有农产品各一个月的价格做出预测，并以此进行可行性和科学性分析。

对于问题四，对提出的“农产品滞销风险评价方法”进行多次实验并且优化，确保该模型得以有效实施。

**关键字：**农产品滞销 NLP 词频 时间序列分析 机器学习 风险预测 价格预测

一、问题重述

1.1问题背景

农产品滞销在我国农村一直是一个大问题，而且国内的农产品滞销几乎快成了年年都发生的大概率事件，农产品滞销潮一波接一波，有效解决农产品滞销问题成了农民和政府的一块心病。由于不了解市场需求，农民的盲目跟风是农村经济发展中一直存在的问题，结果就导致农产品的严重供过于求。目前人们提出的诸如生产前的订单农业方法和出现滞销后借助营销手段的解决方法都只能解一时之急，治标不治本。本文通过设计一个合理的农产品滞销评估风险模型，来对农产品滞销风险进行量化和评估。

1.2问题的提出

（1）在某一农产品滞销前后，各类新闻媒体都会有所涉及该农产品的相关信息，请根据以往曾滞销农产品在新闻媒体上所报道的信息，选出至少4个导致农产品滞销的最主要因素，并说明选择理由。（在选择这些因素时，主要依据的参考内容和数据要可以通过互联网等公开媒体信息查到）

（2）根据上述所选出的关键影响因素，构造出评价农产品滞销风险等级的量化指标和计算公式，并且可以依据所构建的计算公式来评定任何一个农产品在下一生产周期中出现滞销风险大小，进行“很大、大、不大、小、很小”5个等级的评定；并且说明给出这样等级评定的依据。

（3）利用计算公式的评级结果，来评价任何一个农产品下一生产周期中滞销的风险的方法，并结合实际案例来阐述农产品滞销风险评价方法的可行性和科学性。

（4）阐述如何有效实施评价方法，并将其推荐给国家农业部门主管。

1.3解题过程

|  |  |  |
| --- | --- | --- |
| 问题 | 问题目标 | 解决思路 |
| 问题1 | 选出至少4个导致农产品滞销的最主要的因素 | 通过对媒体网络公开的信息数据进行统计分析 |
| 问题2 | 构造出评价农产品滞销风险的量化指标和计算公式 | 利用机器学习-时间序列分析融合方法预测农产品下一个周期内的价格 |
| 对农产品滞销风险进行5个等级的划分 | 对所有农作物价格波动幅度进行归一化处理 |
| 问题3 | 评价任何一个农产品下一生产周期出现滞销的风险 | 针对不同的农产品不断进行实验分析并对模型进行改进优化 |
| 问题4 | 给国家农业部门主管写一份推荐信 | 确定模型应用场景，验证模型准确率 |

1.4相关数据的采集及预处理

由于农产品滞销受到越来越多的媒体关注，如互联网、报纸、新闻等报道，所以相关的信息数据越来越官方且合乎实际；我们所采集的数据主要来源于“新浪舆情”和“中国供销合作社价格指数网”。我们通过编写python爬虫程序从“中国供销合作社价格指数网”、获取了从2017年6月到2018年6月的市场动态及农产品市场价格并从“新浪舆情”获取了从2018年3月19日到2018年6月3日的《农产品滞销数据分析报告》，共计21万余条原始数据，涵盖全国22个省及直辖市（图1）、42个大型农产品供销批发市场（图2），这些数据主要包含了各类农产品所在城市交通密集度，农产品名称，农产品价格，农产品价格日期，农产品种类，农产品省份来源，农产品上市日期及季节，农产品上市天气状况等。我们对文本信息及数据进行预处理之后得到我们所需要的标准化数据。

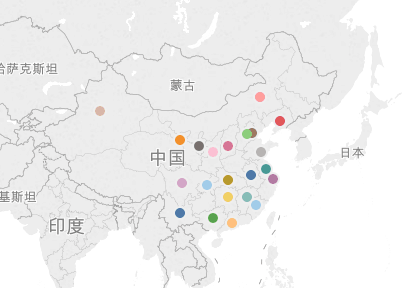
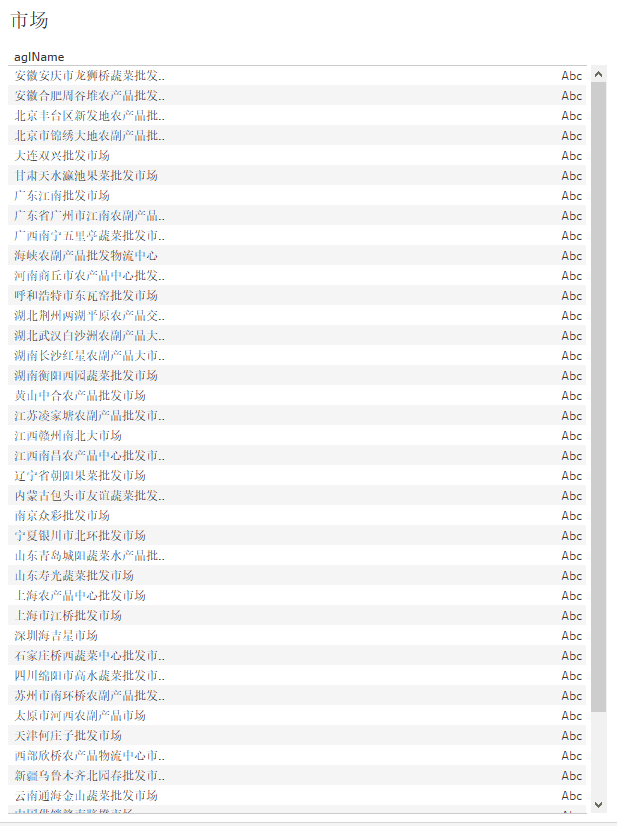
 

图1 数据覆盖22个省及直辖市 图2 数据覆盖42个农产品供销批发市场

二、问题分析

2.1 问题一分析

问题一是要选出至少4个导致农产品滞销的最主要因素，首先，根据互联网“新浪舆情”中的“滞销地图”话题所提供的从2018年3月19日到2018年6月3日的《农产品滞销数据分析报告》， 对其中的农产品滞销的各种原因进行初步的统计分析。其次，再通过对“中国供销合作社”发布的近一年的农产品市场动态信息进行自然语言处理（NLP）的词频分析，得出关于农产品滞销提及率较高的几个因素。

2.2 问题二分析

在问题一的条件下，我们可以得出几个导致农产品滞销的最主要因素。对于这些主要因素进行相关性及可视化分析，接着将收集到的数据进行数值化分析，通过时间序列分析-机器学习融合方法建立模型，即可得出评价农产品滞销风险等级的量化指标和计算公式。

2.3 问题三分析

根据问题二中建立的农产品滞销风险评估模型，我们对所有的农产品进行模拟测试，预测它们一个月的价格情况作为我们模型的评价对象，对我们提出的评价方法进行可行性和科学性的分析。

2.4 问题四分析

在基于问题三的基础上，对提出的“农产品滞销风险评价方法”再进行多次实验并且优化，针对该模型提出有效的应用场景和应用方法。

三、问题一

3.1概念定义

自然语言处理：通过计算机技术对信息文本进行处理，包括文本的检索 ，计算某个词出现的频率，某句话的各个词组进行划分及词性标注等。

词频：针对我们所收集的所有关于农产品滞销的文本信息计算每个词出现的频率。

城市交通密度：用来评价城市道路的发展规模，可借此体现农作物运输的难易程度。

3.2数据来源

从“新浪舆情”获取了从2018年3月19日到2018年6月3日的《农产品滞销数据分析报告》（图3），从“中国供销合作社”价格指数网获取了从2017年6月到2018年6月的市场动态（图4）及农产品市场价格（图5），共计21万余条原始数据，涵盖全国22个省及直辖市、42个大型农产品供销批发市场，这些数据主要包含了各类农产品所在城市交通密集度，农产品名称，农产品价格，农产品价格日期，农产品种类，农产品省份来源，农产品上市日期及季节，农产品上市天气状况等，从中国国家统计局获取了1949年到2011年各省交通运输情况，这些数据包括地区、铁路里程、公路里程、国土面积、交通面积等（图5）。



图3 《农产品滞销数据分析报告》部分数据



图4 “中国供销合作社”市场动态部分文本数据



图4 中国农产品市场价格部分数据

3.3数据分析

对“新浪舆情”的“滞销地图”话题提供的2018年3月19日到2018年6月3日的《农产品滞销数据分析报告》中关于农产品滞销的原因进行了统计分析（图6），可以总结归纳出9种原因，其中天气、市场行情（包括市场价格设定、供求关系、产品出口等）及缺乏销售渠道是影响农产品滞销因素。



图6 农产品滞销的原因统计分析

接着，结合“中国供销网”提供的市场动态、各省的交通运输情况及42个大型农产品批发市场价格进行词频分析（图7），发现农产品的种类及地区也是影响农产品的滞销因素。

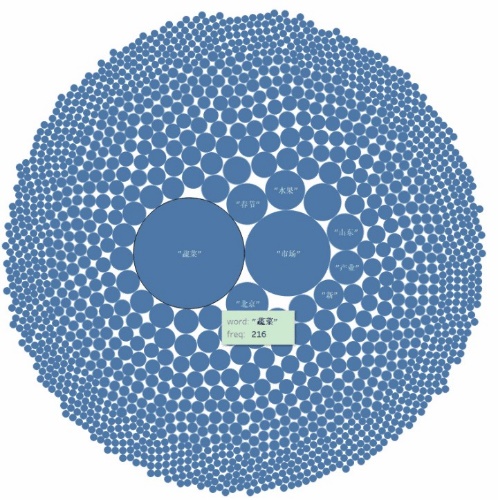
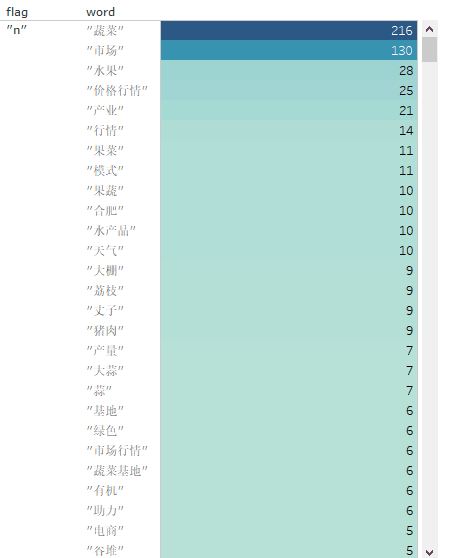


图7 词频分析

3.4结论

多媒体发布关于农产品滞销的信息中某些词出现的频率越高越可以反映出该词的热门程度及重要程度，当提及原因时，则更可以体现其重要性，因此根据互联网上关于市场动态的文本信息的统计分析以及对文本信息的词频信息，我们可以总结出较为官方的影响农产品滞销的因素，再加上对“新浪舆情”发布的《农产品滞销数据分析报告》中所有提出的农产品滞销原因进行分类汇总，因此我们可以得出结论：导致农产品滞销的最主要因素有农产品上市期间的天气、市场行情（包括市场价格、供求关系）、城市交通密度（体现农作物运输的难易程度）、地区、农产品的类型。

四、问题二

4.1概念定义

**熵：**在信息论与概率统计中，熵是表示随机变量不确定性的度量。设X是一个取有限个值的离散随机变量，其概率分布为

则随即变量X的熵定义为

式中的对数以2为底或以e为底（自然对数），这时熵的单位分别称作比特或纳特。由定义可知，熵只依赖于X的分布，而与X的取值无关，所以也可以将X的熵记作H(p)，即熵越大，随机变量的不确定性就越大。

当p=0或p=1时H(p)=0，随机变量完全没有不确定性。当p=0.5时，H(p)=1，熵取值最大，随机变量不确定性最大。设有随机变量(X,Y)，其联合概率分布为：

条件熵H(Y|X)表示在已知随机变量X的条件下随机变量Y的不确定性。随机变量X给定的条件下随机变量Y的条件熵H(Y|X)，定义为X给定条件下Y的条件概率分布的熵对X的数学期望为：

这里，。

当熵和条件熵中的概率由数据估计（特别是极大似然估计）得到时，所对应的熵与条件熵分别成为经验熵和经验条件熵。

**信息增益：**表示得知特征X的信息而使得类Y的信息的不确定性减少的程度。

特征A对训练数集D的信息增益g(D,A),定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即：

一般地,熵H(Y)与条件熵H(Y|X)之差称为互信息,决策树学习中的信息增益等价于训练数据集中类与特征的互信息.

**信息增益的算法：**

输入:训练数据集D和特征A;

输出:特征A对训练数据集D的信息增益g(D|A).

1. 计算数据集D的经验熵H(D)：
2. 计算特征A对数据集D的经验条件熵H(D|A)：
3. 计算信息增益：

**决策回归树的生成：**

假设X与Y分别是输入和输出向量，并且Y是连续变量，给定训练数据集

https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D228/sign=69ca2c23a7345982c18ae29034f4310b/b58f8c5494eef01fc43ea092eafe9925bc317dbb.jpg

一个回归树对应着输入空间（即特征空间）的一个划分以及在划分的但单元上的输出值。假设已将输入空间划分为M个单元，并且在每个单元上有一个固定的输出值，于是回归树模型可表示为：

https://gss0.bdstatic.com/-4o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D163/sign=e27880d4052442a7aa0ef9a3e242ad95/b21c8701a18b87d65b5611480d0828381f30fd55.jpg

当输入空间的划分确定时，可以用平方误差：

https://gss0.bdstatic.com/94o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D122/sign=993eb6f053ee3d6d26c683c971166d41/342ac65c1038534333f122769913b07eca8088af.jpg

来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值。易知，单元上的的最优值是上所有输入实例对于的输出的均值，即：

https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D141/sign=21e1eb560055b31998f9867172a98286/562c11dfa9ec8a13ed05f0b7fd03918fa0ecc04e.jpg

接着采用启发式的方法对输入空间进行划分，选择第j个变量和它取的值s，作为切分变量和切分点，并定义两个区域：

https://gss2.bdstatic.com/9fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D292/sign=e1ae419afd36afc30a0c386c8118eb85/03087bf40ad162d910326e461bdfa9ec8b13cd88.jpg

然后寻找最优切分变量j和最优切分点s。具体的，求解：

https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D423/sign=709d9fa676d98d1072d40d33123eb807/0823dd54564e92588f498fb49682d158ccbf4e6e.jpg

对固定输入变量j可以找到最优切分点s：

https://gss0.bdstatic.com/-4o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D334/sign=ad560840cc5c1038207ec8c18611931c/e4dde71190ef76c6e3e9f3719716fdfaaf5167b4.jpg

遍历所有输入变量，找到最优切分变量j，构成一个对(j，s)，依次将输入空间划分为两个区域。接着，每个对每个区域重复上述划分过程，直到满足停止条件为止。这样就生成一棵回归树，这样的回归树通常被称为最小二乘树。

**最小均方差：**通过模型不断递推迭代知道树不能在分叉为止，此时方差最小，也就是期望最大化，来划分出最优的决策树。

4.2符号说明

|  |  |
| --- | --- |
| 符号 | 意义 |
|  | 18种农产品的每日预测的单价 |
|  | 35个城市离散化结果 |
|  | 城市密度 |
|  | 农产品上市期间的温度 |
|  | 农产品上市期间的月份（1~12月） |
|  | 农产品上市期间的季节（四个季节） |
|  | 三个农产品的种类（蔬菜、水果、肉质类） |
|  | 22个省份 |
|  | 65种农产品 |
| b | 偏置向量（其他影响农产品滞销的因素） |

4.3影响农产品滞销因素的相关性分析

由于农产品的滞销目前没有比较好的量化表示，因此我们通过农产品价格的变化来作为滞销情况的一种衡量因此我们分别对各个可能影响农产品滞销的因素和农产品的价格之间的相关性进行分析。

农产品上市的天气对价格的影响（图8），可以很清晰的看出雪天及浮尘天气对农产品的价格有很大的影响。

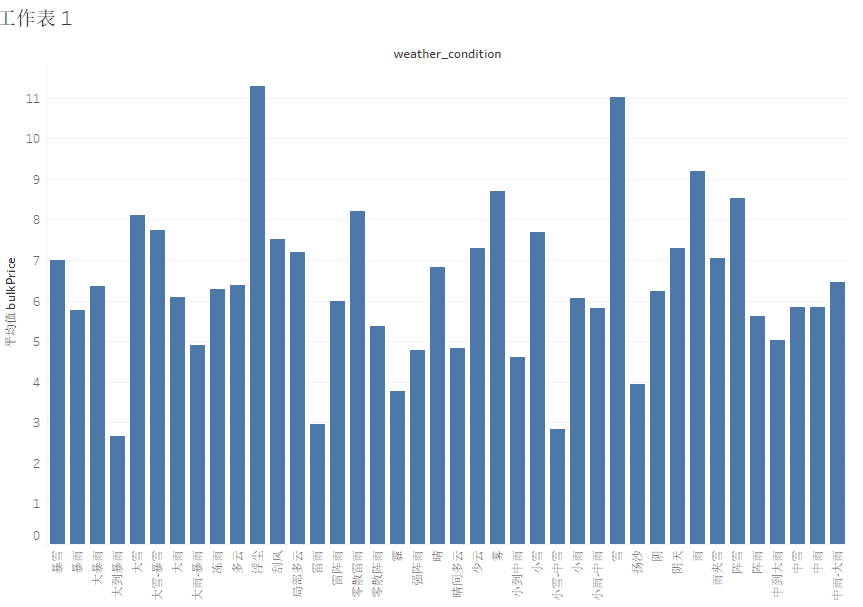


图8 农产品上市的天气对价格的影响

农产品上市的月份对价格的影响（图9），可以看出过年期间菜价会有明显的提高，其他时间价格都比较持平，因此上市月份对价格的影响是比较不明显的。

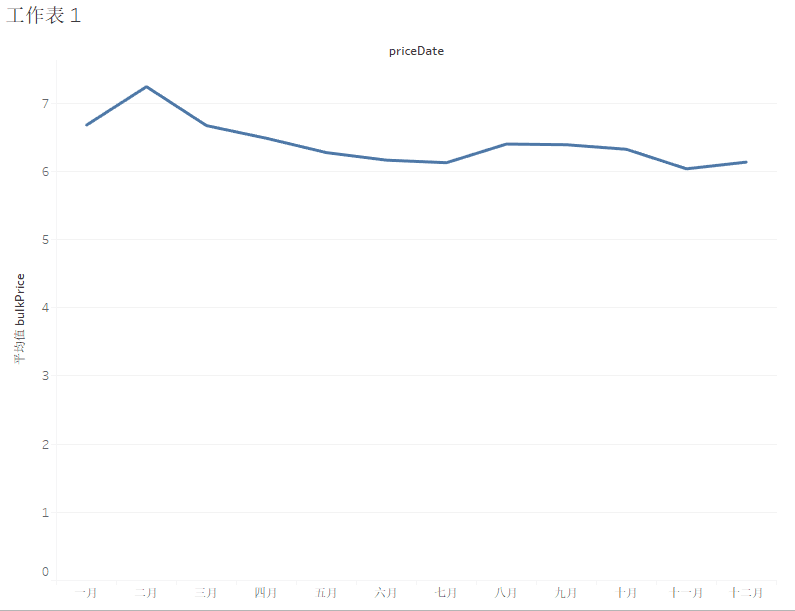


图9 农产品上市的月份对价格的影响

城市交通密度对价格的影响（图10），可以看出不同的城市密度对价格的影响对比是非常明显的；

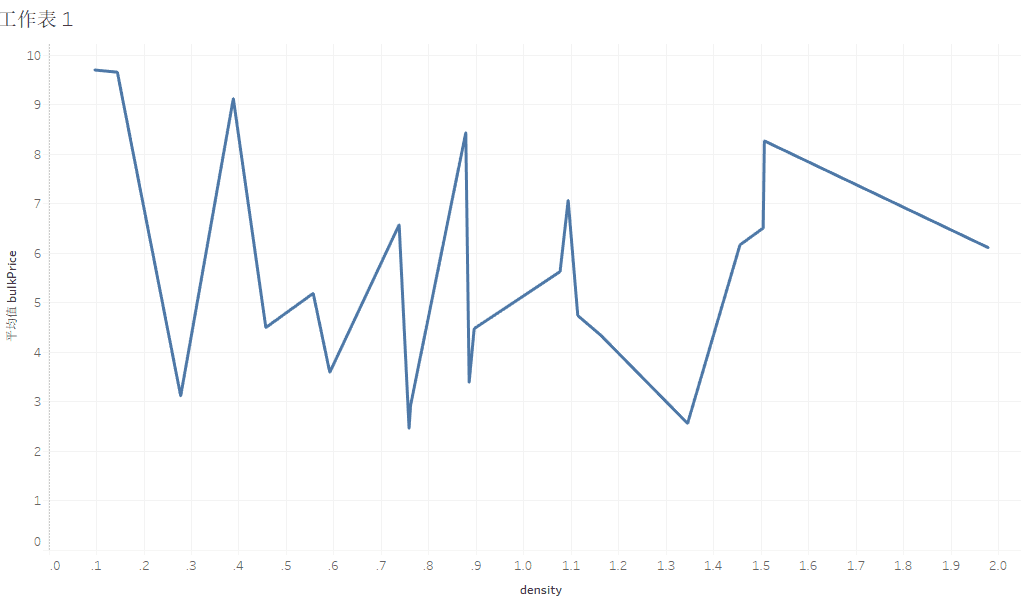


图10 城市交通密度对价格的影响

农产品上市期间的温度对价格的影响（图11）可以看出温度对价格的影响还是有一点的，但不是特别显著。

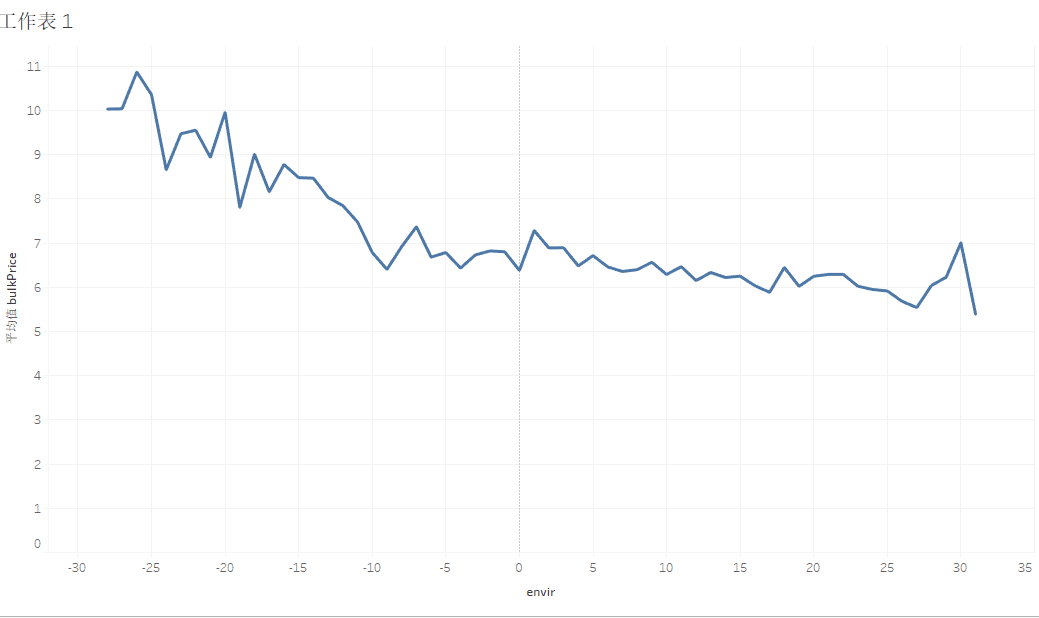


图11 农产品上市的温度对价格的影响

不同类的农产品的平均价格（图12），从图中可以看出，水果类农产品在任一时期内的平均价格变化幅度较小不明显，而蔬菜类农产品则表现为主要受过年期间的影响，价格会突然上涨而随后回落至正常水平，肉禽类农产品则价格变化极为显著，不论是过年期间还是平时都极易受影响，从商品本身来说，农产品自身种类差异也呈现出不同的价格属性变化，像白萝卜这类农产品价格常年无变化。

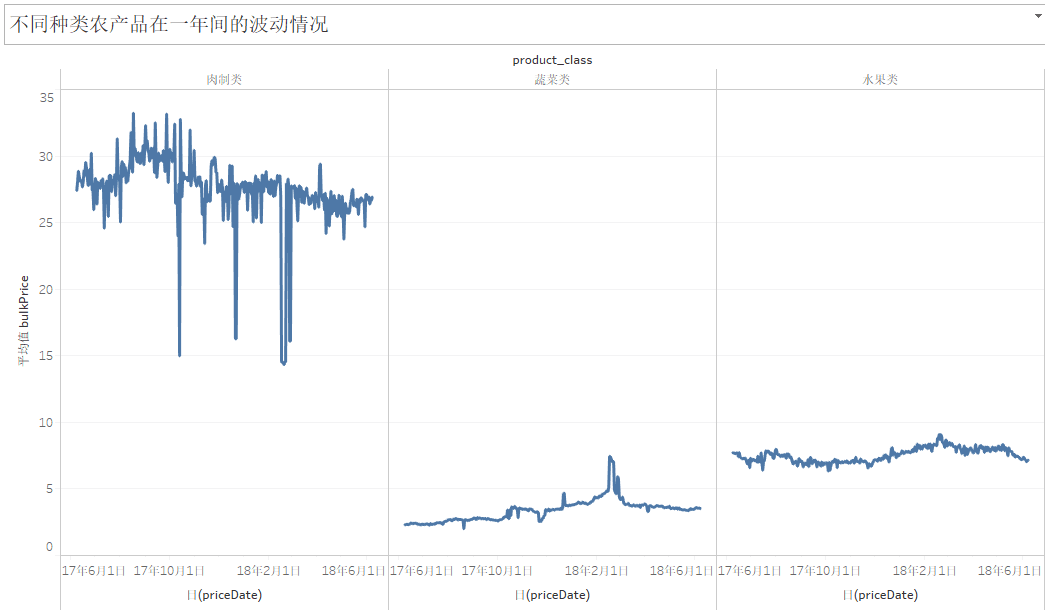


图12（1） 不同种类农产品的平均价格

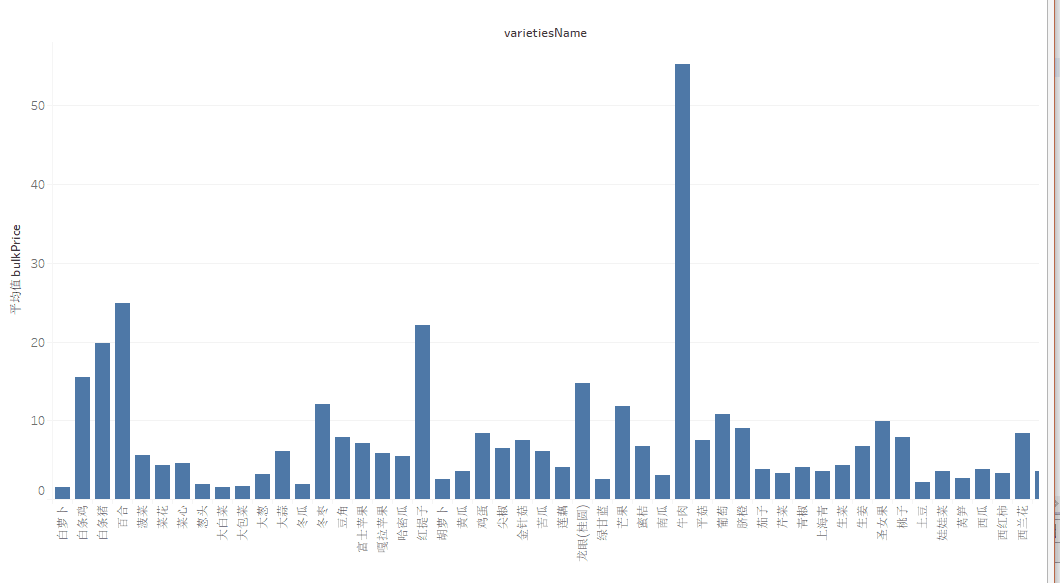


图12（2） 不同种类农产品的平均价格

4.3农产品价格预测模型建立

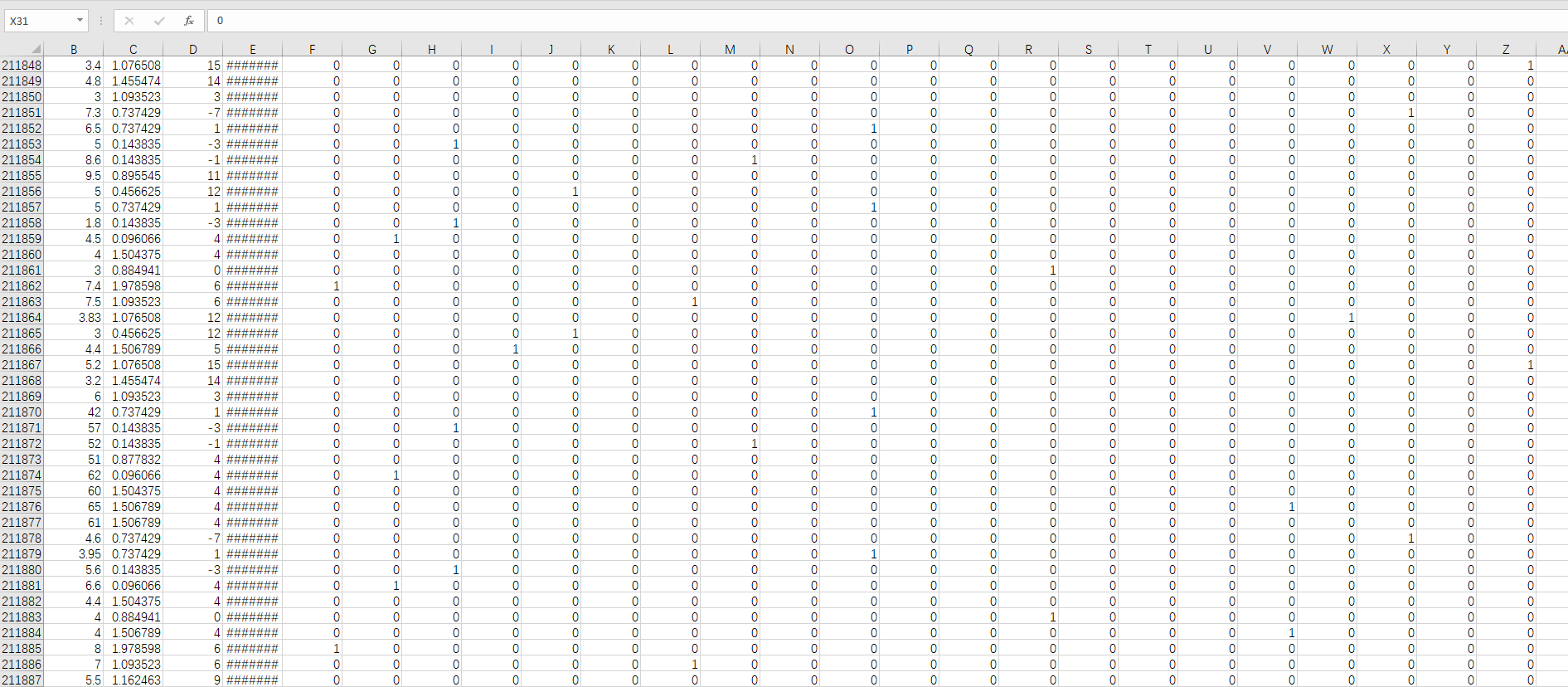
我们的目标是通过建立农产品价格预测模型来对计算农产品在一定时间内的价格波动幅度，并最后根据这个幅度作为农产品滞销风险的等级评判依据。

首先我们要能通过最小均方差来寻找回归决策树的最佳形态，设目标变量y，输入变量，目标变量与输入变量有如下关系：

C:\Users\Hamburger\AppData\Roaming\Tencent\Users\2788808950\TIM\WinTemp\RichOle\D[JFWC~@8(62B($}4LYCXYX.png

上式中的是一个误差项, 表示模型未捕捉的特征或随机噪声，也可以解释为 我们未考虑的因素所形成的的偏置项的一个向量集合。我们最终得到的最小均方差为：

我们将天气，月份，城市等离散型多维数据进行onehot（热编码处理）变为数值型，并具有良好的独立性，对于离散型数据(如温度，密度)进行归一化对连续时间序列，通过划分窗口的形式对数据进行处理，并添加到数据中，形成新的特征。最后得到一个高维矩阵。



21w\*183维的数据。非常稀疏，考虑到使用PCA 进行降维。但是与工业界上的数据相比，并没有太多，直接拿这个矩阵来进行了训练，测试和验证。

4.4农产品风险量化及等级划分模型

我们通过预测模型连续预测几个同等大小时间窗口的上市价格，设第天的市场价格为,时间窗口的大小为天(为奇数)，则从第天后，第个时间窗口内的价格为,计算第个时间窗口内的价格的价格波动及第个时间窗口内的价格(第个时间窗口包含在第个时间窗口内且大小不等于第个时间窗口)的价格波动,则对所有的农产品的滞销风险系数可定义为：，这是对所有农产品的波动情况进行标准化处理（归一化处理），最后将所有的农产品的滞销风险系数进行排序，根据的值以及排序的结果对整个排序范围按每个等级包含相同农产品分布情况进行5个等级的划分，如果排序分布情况均匀，则均匀划分5个等级，若产品分布情况比较分散，则可以根据农产品分布的聚集程度进行等级划分。

1. 问题三

5.1模型的应用

本次建模得到的是一个基于时间序列和机器学习算法模型融合的价格预测的模型，通过从宏观上观察市场农产品价格动态，来评估农产品滞销的风险。从本质上上，价格的预测已经是对农产品风险评估最好的量化方式，通过预测价格，我们可以通过多个时间窗口来推测滞销的可能性。在本次建模实验中，我们用了机器学习算法中的多个模型和时间序列进行融合来对原始数据进行拟合包括:（DecisionTreeRegressor，LinearRegression，随机森林，adaboost回归，XGBRegressor），并对部分验证集进行验证，从以下四个指标来评价及进行了验证。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | DR | LR | RF | AR | XGB |
| MAE | 0.76 | 119167380142.0 | 0.72 | 8.95 | 1.39 |
| MSE | 1.94 | 5.61714600186e+22 | 1.65 | 87.21 | 4.63 |
| MAE | 0.4 | 1.36 | 0.42 | 9.62 | 0.96 |
| EVS | 0.98 | -4.50273076297e+20 | 0.98 | 0.86 | 0.95 |

MAE：平均绝对值误差（mean absolute error）

MSE: 均方误差（mean squared error）

MAE：中位数绝对值误差（median absolute error）

EVS：解释方差评分（explained variance score） 越小代表预测结果越差

以上四个评价标准，代表对回归模型的拟合程度。本次建模将数据集划分为训练集，测试集和验证集。训练加测试的数据为5月之前所有的数据，验证集为5月的数据。后附各个模型的验证结果与真实结果的比对信息。

5.2模型结果与分析

通过验证结果分析，我们发现决策树回归对于此次预测任务具有较好的评价结果，备受青睐的XGboost也表现一般，且具有较大的误差。这说明，在模型中有些重要的特征具有较大的权重，我们可以通过剪枝算法，来对树的规模进行裁剪。同时考虑到叶子节点的决策边界是，可以考虑使用除了lsd以外的优化函数。另外从上边训练模型中可以看出，随机森林具有更好的评价效果，但是与决策树回归得到的效果不差太多，且决策树更简单，所以此次比赛选用决策树回归优化算法作为我们的回归模型。

5.3模型的可行性与科学性

可行性和科学性分析:

本次建模都采用的历史数据,包括（天气，温度，交通复杂度，地点，市场价格行情）等公开信息，较容易获取，成本几乎没有，经济上可行。

本次建模只采用了较少的特征信息，但是较好的拟合了未来价格趋势，时间充足的情况下能找到更多的特征信息，从而达到技术可行。

本次的建模算法采用的是决策树回归所使用的优化算法，它结合了信息熵，计算机科学都多个领域的知识,在数学上得到理论上的实证。具有相当强的科学性。通过预测结果，我们就能发现以决策树作为基础模型，都具有较好的评价效果，比如随机森林。

六、问题四

尊敬的国家农业部主管：

您好！

我们是某某高校研究生，鉴于农产品滞销状况越来越受到社会各界的关注，我们也希望能够借助一定的科学方法来帮助解决农产品滞销问题，甚至能够预测滞销风险。

日前，我们通过“新浪舆情”、“中国供销合作社价格指数网”等较为官方的信息渠道，收集了当前经济环境下，自2017年6月起至2018年6月近一年的各类农产品基础信息和滞销信息（包括各类农产品所在城市交通密集度，农产品名称，农产品价格，农产品价格日期，农产品种类，农产品省份来源，农产品上市日期及季节，农产品上市天气状况等）。根据收集的信息进行详细的分析与处理。

由于多媒体发布关于农产品滞销的信息中某些词出现的频率越高越可以反映出该词的热门程度及重要程度，当提及原因时，则更可以体现其重要性，因此根据互联网上关于市场动态的文本信息的统计分析以及对文本信息的词频信息，我们总结出了较为官方的影响农产品滞销的因素，再加上对“新浪舆情”发布的《农产品滞销数据分析报告》中所有提出的农产品滞销原因进行分类汇总，我们认为导致农产品滞销的最主要因素有农产品上市期间的天气、市场行情（包括市场价格、供求关系）、城市交通密度（体现农作物运输的难易程度）、地区、农产品的类型。

并通过构建农产品滞销风险评估模型，来试图对某一农产品将来生产周期中可能出现滞销情况的风险性进行评估，进而借此来帮助农民规避滞销风险，并做应对滞销风险的措施。通过一系列检验，该模型理论上是可行的。

我们根据模型的应用，得出了模型预测的结果比较理想且符合实际，正确率达到：，故，在此斗胆向您推荐我们所建立的滞销风险评估模型将其运用于市场，我们将从以下几个方面来确保该模型的有效实施。

第一，能较好的预测未来时间的农产品价格走向

第二，在一段时间窗口上能够合理的评估农产品所面临了的滞销风险。

第三，不断优化模型，进一步提高预测精度。

如果能够做到以上几点，我相信农产品滞销风险评估能够得到合理运行，并帮助改善当下状况。

最后，希望农产品滞销问题能够早日得到改善！！！

此致

敬礼！

七．模型的评价

6.1 模型的优点

（1）本文根据决策树回归-时间序列分析融合模型能够较为准确预测任一农产品一年以内任何时期的价格，进而根据价格的波动情况对农产品的滞销风险进行预测，且预测结果比较贴近事实。

（2）模型在考虑几个主要因素的情况下，为了防止数据过拟合，通过偏置向量把其他影响较小的影响因素也考虑进去了，减小了模型的误差，增强了模型的可用性。

6.2 模型的缺点

（1）人为地从官方资料中采集各类农产品的相关数据、信息，具有一定的主观性，不够客观。

（2）没有考虑较多主观因素，但是也发现这些足够引起滞销风险。

（3）风险等级评估方法不够完善。

6.3 模型改进

（1）采用其他机器学习算法，比如深度学习，不断尝试调整参数，进一步优化模型，开始用的纯机器学习模型，然后尝试使用时间序列分析与机器学习模型融合，具有较好的拟合能力。

八、参考文献

[1]陈祥云.渠道阻碍视角的生鲜农产品滞销预测及疏通策略研究[D].重庆交通大学,2016.

[2]汤淼淼.安徽省农产品滞销关键环节及原因实证研究[D].安徽农业大学,2014.

[3]周玲,韩朝怡.基于机器学习建模方法的我国粮食种植面积影响因素研究[J].时代金融,2017(11):324+334.

[4]复杂数据统计方法[M].中国人民大学出版社,吴喜之,2015

[5]统计学习方法[M].清华大学出版社,李航,2012

九、附录

问题一源程序（数据采集）：

(价格数据采集程序)price\_data\_collect.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月8日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

'''

from bs4 import BeautifulSoup

from datetime import datetime

import threading

import requests

import urllib.request;

import time

import random

import json

import re

import redis

import pymysql

from pymongo import MongoClient

import DBUtils

from DBUtils.PooledDB import PooledDB

user\_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'

headers={'User-Agent':user\_agent}

#pool = PooledDB(pymysql,5,host='localhost',user='root',passwd='147258',db='math\_model',port=3306,charset='utf8mb4') #5为连接池里的最少连接数

#conn = pool.connection() #以后每次需要数据库连接就是用connection（）函数获取连接就好了

import re

def price\_collect(url):

html = urllib.request.urlopen(url).read().decode('utf8')

html=html[43:-32]

data=json.loads(html)#json的数据格式

for i in range(len(data)):

priceDate=data[i]['priceDate']#产品日期

varietiesName=data[i]['varietiesName']#产品名

bulkPrice=data[i]['bulkPrice']#产品价格

aglName=data[i]['aglName']#售卖市场

connectMysql(conn, priceDate, varietiesName, bulkPrice, aglName)

def connectMysql(connection,priceDate,varietiesName,bulkPrice,aglName):#连接数据库并插入数据

#获取会话指针

with connection.cursor() as cursor:

#创建sql语句

sql = "insert into `priceDetails` (`priceDate`,`varietiesName`,`bulkPrice`,`aglName`) values (%s,%s,%s,%s)"

#执行sql语句

cursor.execute(sql,(priceDate,varietiesName,bulkPrice,aglName))

#提交数据库

connection.commit()

if \_\_name\_\_ == '\_\_main\_\_':

conn = pymysql.connect(host='127.0.0.1', user='root', password='147258', db='math\_model', charset='utf8mb4')

url="http://www.chinaapm-data.com/pq/findByPage?draw=2&columns%5B0%5D%5Bdata%5D=varietiesName&columns%5B0%5D%5Bname%5D=&columns%5B0%5D%5Bsearchable%5D=true&columns%5B0%5D%5Borderable%5D=false&columns%5B0%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B0%5D%5Bsearch%5D%5Bregex%5D=false&columns%5B1%5D%5Bdata%5D=priceDate&columns%5B1%5D%5Bname%5D=&columns%5B1%5D%5Bsearchable%5D=true&columns%5B1%5D%5Borderable%5D=false&columns%5B1%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B1%5D%5Bsearch%5D%5Bregex%5D=false&columns%5B2%5D%5Bdata%5D=bulkPrice&columns%5B2%5D%5Bname%5D=&columns%5B2%5D%5Bsearchable%5D=true&columns%5B2%5D%5Borderable%5D=false&columns%5B2%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B2%5D%5Bsearch%5D%5Bregex%5D=false&columns%5B3%5D%5Bdata%5D=aglName&columns%5B3%5D%5Bname%5D=&columns%5B3%5D%5Bsearchable%5D=true&columns%5B3%5D%5Borderable%5D=false&columns%5B3%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B3%5D%5Bsearch%5D%5Bregex%5D=false&columns%5B4%5D%5Bdata%5D=trendUri&columns%5B4%5D%5Bname%5D=&columns%5B4%5D%5Bsearchable%5D=true&columns%5B4%5D%5Borderable%5D=false&columns%5B4%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B4%5D%5Bsearch%5D%5Bregex%5D=false&start=0&length=233811&search%5Bvalue%5D=&search%5Bregex%5D=false&startTime=2017-06-08&endTime=2018-06-08&marketCode=&varietiesCode=&province=&typeCode=&\_=1528469159849"#构造url链接

price\_collect(url)

conn.close()

（历史天气采集程序）history\_weather\_collect.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月9日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

'''

from bs4 import BeautifulSoup

import re

import requests

from pymongo import MongoClient

citylist=['anqing','baotou','changsha','changzhou','chaoyang','dalian','fuzhou','ganzhou','guangzhou','hefei','hengyang','huhehaote','huangshan','jingzhou','mianyang','nanchang','nanjing','nanning','qingdao','shangqiu','shanghai','shenzhen','shijiazhuang','shouguang','suzhou','taiyuan','taizhou','tianjin','tianshui','wulumuqi','wuhan','xian','xuancheng','yinchuan','yuxi','chongqing']

monthlist=['201707','201708','201709','201710','201711','201712','201801','201802','201803','201804','201805']

con=MongoClient('localhost', 27017)

db=con.modelData

collection=db.weatherData

for j in monthlist:

for i in range(len(citylist)):

city=citylist[i-1]

url='http://www.tianqihoubao.com/lishi/'+citylist[i-1]+'/month/'+str(j)+'.html'

user\_agent = 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4882.400 QQ'

headers={'User-Agent':user\_agent}

r = requests.get(url,headers=headers)

soup = BeautifulSoup(r.text,'html5lib')#html.parser

#print (soup)

tr=soup.find\_all('tr')[8]

td=tr.find\_all('td')

print(td)

priceDate=td[0].text.strip()#日期

weather\_condition=td[1].text.strip()#天气状况

parttern=re.compile('(.\*)℃')

envir =parttern.findall(td[2].text.strip())[1]#

print(envir)

weatherDetails={"priceDate":priceDate,"weather\_condition":weather\_condition,"envir":envir,'city':city}

#collection.insert(weatherDetails)

# for idx, tr in enumerate(soup.find\_all('tr')):

# if idx != 0:

# td=tr.find\_all('td')

# print(td)

# priceDate=td[0].text.strip()#日期

# weather\_condition=td[1].text.strip()#天气状况

# parttern=re.compile('(.\*)℃')

# try:

# envir =parttern.findall(td[2].text.strip())[1]#

# print(envir)

# except IndexError as e:

# continue

#

# weatherDetails={"priceDate":priceDate,"weather\_condition":weather\_condition,"envir":envir,'city':city}

# collection.insert(weatherDetails)

（市场动态信息采集程序）topicWord\_data\_collect.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月8日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

'''

from bs4 import BeautifulSoup

from datetime import datetime

import threading

import requests

import urllib.request;

import time

import random

import json

import re

import redis

import pymysql

from pymongo import MongoClient

import DBUtils

from DBUtils.PooledDB import PooledDB

user\_agent = 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4882.400 QQ'

headers={'User-Agent':user\_agent}

url="http://www.chinaapm-data.com/mn/md"

#pool = PooledDB(pymysql,5,host='localhost',user='root',passwd='147258',db='math\_model',port=3306,charset='utf8mb4') #5为连接池里的最少连接数

con=MongoClient('localhost', 27017)

db=con.modelData

collection=db.marketData

data = open("marketDy.json")

data=json.load(data)

print(data)

import re

def collect\_data(data):

for i in range(len(data)):

publishUser=data[i]["publishUser"]

publishAt=data[i]["publishAt"]

#articleTypeId=data[i]["articleTypeId"]

id=data[i]["id"]

title=data[i]["title"]

url=r'http://www.chinaapm-data.com/mn/mdc/'+str(id)

article=article\_collect(url)

marketDetails={"publishUser":publishUser,"publishAtTime":publishAt,"id":id,"title":title,"article":article}

collection.insert(marketDetails)

def article\_collect(url):

r = requests.get(url,headers=headers)

soup = BeautifulSoup(r.text,'html.parser')#html.parser

print(soup)

data=soup.find(class\_='content3').find\_all('p')

pattern = re.compile(r'<p>(.\*)</p>')

data=pattern.findall(str(data))

print(data)

data=''.join(data)

return data

if \_\_name\_\_ == '\_\_main\_\_':

collect\_data(data)

（自然语言处理词频处理与统计）market\_report\_word\_process.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月10日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

'''

import pandas as pd

import jieba

import jieba,math

import jieba.analyse

import jieba.posseg as psg

import re

from pymongo import MongoClient

con=MongoClient('localhost', 27017)

db = con.modelData

collection=db.wordDetail

def stopwordslist(filepath):

stopwords = [line.strip() for line in open(filepath, 'rb').readlines()]

return stopwords

def seg\_sentence(sentence):

sentence\_seged = psg.cut(sentence.strip())

stopwords = stopwordslist('哈工大停用词表.txt') # 这里加载停用词的路径

outstr = ''

word\_freq = {}

for ele in sentence\_seged:

if ele.word not in stopwords:

if ele in word\_freq:

word\_freq[ele] += 1

else:

word\_freq[ele] = 1

freq\_word = []

for ele, freq in word\_freq.items():

freq\_word.append((ele.word, ele.flag,freq))

freq\_word.sort(key = lambda x: x[1], reverse = True)

for ele.word, ele.flag,freq in freq\_word:

word=ele.word;

flag=ele.flag

freq=freq;

wordDetail = {"word":word,"flag":flag,"freq":freq}

collection.insert(wordDetail)

if \_\_name\_\_ == '\_\_main\_\_':

article=pd.read\_csv(r'C:\Users\Agnostic\Desktop\marketData.csv',encoding='utf8')

words=''

print(article['title'])

for title in article['title']:

if(':' in str(title)):

title=title.replace(':','')

words+=str(title)

print(words)

seg\_sentence(words)

（通用特征处理）data\_general\_process.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月9日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

数据合并

'''

import pandas as pd

import re

from docopt import Pattern

#from data\_collect\_process.history\_weather\_collect import parttern

#pricedata=pd.read\_csv(r'C:\Users\Agnostic\Desktop\pricedetails.csv')

#varietiesName=pd.DataFrame(pricedata)['varietiesName'].drop\_duplicates()

#varietiesName.to\_csv('varietiesName.csv')

def process\_condition():

weather\_condition=weatherdata['weather\_condition']

list=[]

print(weather\_condition[1])#处理天气

for i in weather\_condition:

print(i)

i=i.replace(' ','').replace('\n','').replace('\r','')

parttern=re.compile(r'^(.\*)\/')

print(parttern.findall(i))

i=parttern.findall(i).strip()

print (i)

i=''.join(i)

list.append(i)

weatherdata['weather\_condition']=list

print(weatherdata['weather\_condition'])

weatherdata.to\_csv('weather.csv')

def date\_process(): #日期处理

date=weatherdata['priceDate']

print(date)

parttern1=re.compile(r'^(.\*)年')

parttern2=re.compile(r'年(.\*)月')

parttern3=re.compile(r'月(.\*)日')

for date in date:

year=''.join(parttern1.findall(date))

month=''.join(parttern2.findall(date))

if ('0' in month):

month=month.replace('0','')

day=''.join(parttern3.findall(date))

if ('0' == day[0]):

day[0]=''

#print(year+'/'+month+'/'+day)

list.append(year+'/'+month+'/'+day)

def condition\_reprocess():

list=[]

conditions=weatherdata['weather\_condition']

for i in conditions:

i= i.replace("['",'').replace("']",'')

print (i)

list.append(i)

weatherdata['weather\_condition']=list

weatherdata.to\_csv('weather.csv')

def weather\_price\_process():

weather\_price1=weather\_price.dropna()

weather\_price1.to\_csv('weather\_price.csv')

def weather\_10\_process():

list=[]

for i in weather\_10\_process1['priceDate']:

a= i.replace('2017/1','2017/10')

list.append(a)

weather\_10\_process1['priceDate']=list

weather\_10\_process1.to\_csv('weatehr\_patch.csv')

if \_\_name\_\_ == '\_\_main\_\_':

list=[]

weatherdata=pd.read\_csv(r'C:\Users\Agnostic\Desktop\weather.csv',encoding='utf8')

weather\_price=pd.read\_csv(r'C:\Users\Agnostic\Desktop\preprocess\weather\_price.csv',encoding='utf8')

weather\_10\_process1=pd.read\_csv(r'weather\_patch.csv')

#date\_process()

#process\_condition()

#condition\_reprocess()

weather\_price\_process()

#weather\_10\_process()

# weather=pd.read\_csv('weather.csv')

#print(weather)

（时间序列程序）transport\_price.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月11日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

'''

from pandas import DataFrame

from pandas import concat

import pandas as pd

time\_ser\_process=pd.read\_csv('pricedetails\_plus.csv')#准备进行时间序列处理

time\_ser\_process=time\_ser\_process.dropna(axis=0)

print(time\_ser\_process['citys'])

time\_ser\_process.to\_csv('pricedetails\_plus\_1.csv')

def series\_to\_supervised(data, n\_in=1, n\_out=1, dropnan=True):#时间序列函数

"""

Frame a time series as a supervised learning dataset.

Arguments:

data: Sequence of observations as a list or NumPy array.

n\_in: Number of lag observations as input (X).

n\_out: Number of observations as output (y).

dropnan: Boolean whether or not to drop rows with NaN values.

Returns:

Pandas DataFrame of series framed for supervised learning.

"""

n\_vars = 1 if type(data) is list else data.shape[1]

df = DataFrame(data)

cols, names = list(), list()

# input sequence (t-n, ... t-1)

for i in range(n\_in, 0, -1):

cols.append(df.shift(i))

names += [('var%d(t-%d)' % (j+1, i)) for j in range(n\_vars)]

# forecast sequence (t, t+1, ... t+n)

for i in range(0, n\_out):

cols.append(df.shift(-i))

if i == 0:

names += [('var%d(t)' % (j+1)) for j in range(n\_vars)]

else:

names += [('var%d(t+%d)' % (j+1, i)) for j in range(n\_vars)]

# put it all together

agg = concat(cols, axis=1)

agg.columns = names

# drop rows with NaN values

if dropnan:

agg.dropna(inplace=True)

return agg

#print(time\_ser\_process.describe())

algname=time\_ser\_process['aglName'].drop\_duplicates()

varietiesName=time\_ser\_process['varietiesName'].drop\_duplicates()

time\_ser\_process['priceDate']=pd.to\_datetime(time\_ser\_process['priceDate'])

#print(algname)

df = pd.DataFrame(columns = ["aglName", "bulkPrice ", "citys", "envir",'priceDate','product\_class','season ','varietiesName','weather\_condition','bulkPrice\_day7','bulkPrice\_day6','bulkPrice\_day5','bulkPrice\_day4','bulkPrice\_day3','bulkPrice\_day2','bulkPrice\_day1',])

for i in list(algname):

#print(i)

data= time\_ser\_process[time\_ser\_process['aglName']==i]

#print(data)

data\_1=data['varietiesName'].drop\_duplicates()

#print(data\_1)

for j in varietiesName:

print(j)

if(j in list(data\_1)):

data1=data[data['varietiesName']==str(j)]

#print(data1['priceDate'])

#data1['priceDate']=pd.to\_datetime(data1['priceDate'])

#data1=data1.sort\_index(inplace=True)

data1=data1.sort\_values('priceDate')

bulkprice=list(data1['bulkPrice'])

#print(bulkprice)

bulkprice=series\_to\_supervised(bulkprice, n\_in=7, n\_out=1, dropnan=False)

bulkprice=pd.DataFrame(bulkprice)

#print(bulkprice)

bulkprice.columns=['bulkPrice\_day7','bulkPrice\_day6','bulkPrice\_day5','bulkPrice\_day4','bulkPrice\_day3','bulkPrice\_day2','bulkPrice\_day1','bulkPrice']

#print(bulkprice)

#print(bulkprice)

#bulkprice=pd.DataFrame(bulkprice,columns=['bulkPrice\_day7','bulkPrice\_day6','bulkPrice\_day5','bulkPrice\_day4','bulkPrice\_day3','bulkPrice\_day2','bulkPrice\_day1','bulkPrice'])

#print(bulkprice)

bulkprice=bulkprice[['bulkPrice\_day7','bulkPrice\_day6','bulkPrice\_day5','bulkPrice\_day4','bulkPrice\_day3','bulkPrice\_day2','bulkPrice\_day1']]

#data1=data1.append(bulkprice)

data1=pd.concat([data1,bulkprice],axis=1)

df=df.append(data1)

print(df)

else :

continue

df.to\_csv(r'data1.csv')

（模型建立）model.py

#!/home/admin/anaconda3/envs/TF/bin/ python3.5

# -\*- coding: utf-8 -\*-

'''

Created on 2018年6月11日

**@author:** Zhukun Luo

Jiangxi university of finance and economics

'''

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import sklearn.metrics as sm

from sklearn.preprocessing import OneHotEncoder#onehot对独立型特征进行编码

from sklearn.linear\_model import LassoCV

from sklearn.cross\_validation import train\_test\_split

from tpot import TPOTRegressor

from xgboost import XGBRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.svm import SVR

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import BaggingRegressor

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

from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor

from mlxtend.regressor import StackingRegressor

import matplotlib.pyplot as pl

from sklearn.linear\_model import MultiTaskLasso, Lasso

from sklearn.cross\_validation import cross\_val\_score

from sklearn.linear\_model import LassoCV

dataset=pd.read\_csv(r'weather\_price\_density.csv')

#print(dataset.isnull())

dataset=pd.get\_dummies(dataset,columns=['city','month','product\_class','province','season','varietiesName','weather\_condition'])

#print(dataset)

#dataset.to\_csv(r'C:\Users\Agnostic\Desktop\dataset\_onehot.csv',encoding='utf8')

dataset['priceDate']=pd.to\_datetime(dataset['priceDate'])

#print(dataset.head(10))

#print(dataset[dataset['priceDate']=='2017-12'].head(10))

use\_set=pd.read\_csv(r'dataset\_onehot\_use\_set.csv')

validate\_set=pd.read\_csv(r'dataset\_onehot\_validate\_set.csv')

predict\_y=validate\_set['bulkPrice']

predict\_x=validate\_set.drop(columns=['bulkPrice'])

#print(use\_set.describe())

target=use\_set['bulkPrice']

print(target)

train\_data=use\_set.drop(columns=['bulkPrice'])

#print(train\_data.head(10))

train\_X, test\_X, train\_y, test\_y = train\_test\_split(train\_data, #划分训练集

target,

test\_size = 0.1,

random\_state = 0)

#params = [1]

#test\_scores = []

#for param in params:

#clf = XGBRegressor( learning\_rate=0.1, max\_depth=param, min\_child\_weight=2)#效果一搬 #56%max\_depth=1

# clf=GradientBoostingRegressor(loss='ls', alpha=0.9,#40%

# n\_estimators=500,

# learning\_rate=0.05,

# max\_depth=1,

# subsample=0.8,

# min\_samples\_split=9,

# max\_leaf\_nodes=10)#这方法也不行

#model\_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(train\_X, train\_y) # 此处 alpha 为通常值 #fit 把数据套进模型里跑

# lr = LinearRegression()#效果不好不到50%

# rfg = RandomForestRegressor(bootstrap=True, max\_features=0.005, min\_samples\_leaf=11, min\_samples\_split=10,

# n\_estimators=100)#最高59%

#

# svr\_rbf = SVR(kernel='rbf')

# clf=svr\_rbf#跑不出来。。。我擦

# stregr = StackingRegressor(regressors=[las, las, las, las], meta\_regressor=svr\_rbf)

# test\_score = np.sqrt(-cross\_val\_score(las, test\_X, test\_y, cv=10, scoring='neg\_mean\_squared\_error'))#70%的正确率,与lasso差不多

# print(test\_score)

#pl.plot(params, test\_scores)

#pl.title("max\_depth vs CV Error");

#pl.show()

#此路不通

###########2.回归部分##########

def try\_different\_method(model):

model.fit(train\_X,train\_y)

# 打印最优的α值

#print ("最优的alpha值: "+str(lassocv.alpha\_.astype('float')))

# 打印模型的系数

#print (lassocv.intercept\_)

#print (lassocv.coef\_)

#score = model.score(test\_X, test\_y)

result = model.predict(predict\_x)

print('mean absolute error=',round(sm.mean\_absolute\_error(predict\_y,result),2))

print('mean squared error=',round(sm.mean\_squared\_error(predict\_y,result),2))

print('median absolute error=',round(sm.median\_absolute\_error(predict\_y,result),2))

print('explained variance score=',round(sm.explained\_variance\_score(predict\_y,result),2))

model\_compair\_data=pd.DataFrame(columns=['predict\_y','true\_y'])

model\_compair\_data['predict\_y']=result

model\_compair\_data['true\_y']=predict\_y

model\_compair\_data.to\_csv('XGB\_compair.csv')#存储真实值与预测值比对结果集

#print('R2 score=',round(sm.r2\_score(test\_y,result),2)

# plt.figure()

# plt.plot(np.arange(len(result)), test\_y,'go-',label='true value')

# plt.plot(np.arange(len(result)),result,'ro-',label='predict value')

# plt.title('score: %f'%score)

# plt.legend()

# plt.show()

# score1 = model.score(predict\_x, predict\_y)

# result1 = model.predict(predict\_x)

# plt.figure()

# plt.plot(np.arange(len(result1)), predict\_y,'go-',label='true value')

# plt.plot(np.arange(len(result1)),result1,'ro-',label='predict value')

# plt.title('score: %f'%score1)

# plt.legend()

# plt.show()

#print(cross\_val\_score(model, train\_X, train\_y, cv=10))

def get\_index():

pass

from sklearn import tree

model\_DecisionTreeRegressor = tree.DecisionTreeRegressor()

####3.2线性回归####

from sklearn import linear\_model

model\_LinearRegression = linear\_model.LinearRegression()

####3.3SVM回归####

from sklearn import svm

model\_SVR = svm.SVR()

####3.4KNN回归####

from sklearn import neighbors

model\_KNeighborsRegressor = neighbors.KNeighborsRegressor()

####3.5随机森林回归####

from sklearn import ensemble

model\_RandomForestRegressor = ensemble.RandomForestRegressor(n\_estimators=20)#这里使用20个决策树,也不错

####3.6Adaboost回归####

from sklearn import ensemble

model\_AdaBoostRegressor = ensemble.AdaBoostRegressor(n\_estimators=50)#这里使用50个决策树，效果也不错

####3.7GBRT回归####

from sklearn import ensemble

model\_GradientBoostingRegressor = ensemble.GradientBoostingRegressor(n\_estimators=100)#这里使用100个决策树

####3.8Bagging回归####

from sklearn.ensemble import BaggingRegressor

model\_BaggingRegressor = BaggingRegressor()

####3.9ExtraTree极端随机树回归####

from sklearn.tree import ExtraTreeRegressor

model\_ExtraTreeRegressor = ExtraTreeRegressor()

from xgboost import XGBRegressor

model\_XgbRegressor=XGBRegressor()

lassocv = LassoCV(alphas=[0.01, 0.1, 0.5, 1, 3, 5, 7, 10, 20, 100], cv=5)

###########4.具体方法调用部分##########

try\_different\_method(model\_XgbRegressor)

#regr\_1 = tree.DecisionTreeRegressor(max\_depth=2)#lsd作为衡量标准

#regr\_2 = tree.DecisionTreeRegressor(max\_depth=5)

#regr\_3 = tree.DecisionTreeRegressor(max\_depth=8)

#regr\_1.fit(train\_X, train\_y)

#regr\_2.fit(train\_X, train\_y)

#regr\_3.fit(train\_X, train\_y)

# Predict

#X\_test = np.arange(0.0, 10.0, 0.01)[:, np.newaxis]

#y\_1 = regr\_1.predict(test\_X)

#y\_2 = regr\_2.predict(test\_X)

#y\_3 = regr\_3.predict(test\_X)

# Plot the results

#plt.figure()

#plt.scatter(train\_X, train\_y, s=20, edgecolor="black",

#c="darkorange", label="data")

#plt.plot(test\_X, y\_1, color="cornflowerblue",

#label="max\_depth=2", linewidth=2)

# plt.plot(test\_X, y\_2, color="yellowgreen", label="max\_depth=5", linewidth=2)

# plt.plot(test\_X, y\_3, color="r", label="max\_depth=8", linewidth=2)

#plt.xlabel("data")

#plt.ylabel("target")

#plt.title("Decision Tree Regression")

#plt.legend()