Q2 分析报告

(一) 问题描述

特征: 指数据中抽取出来的对结果预测有用的信息, 也就是数据的相关属性。

特征工程: 使用专业背景知识和技巧处理数据, 使得特征能在机器学习算法上发挥更好的作用的过程, 在本次作业中特征工程的目的是为后续的模型建立、训练和回归预测提供特征数据。

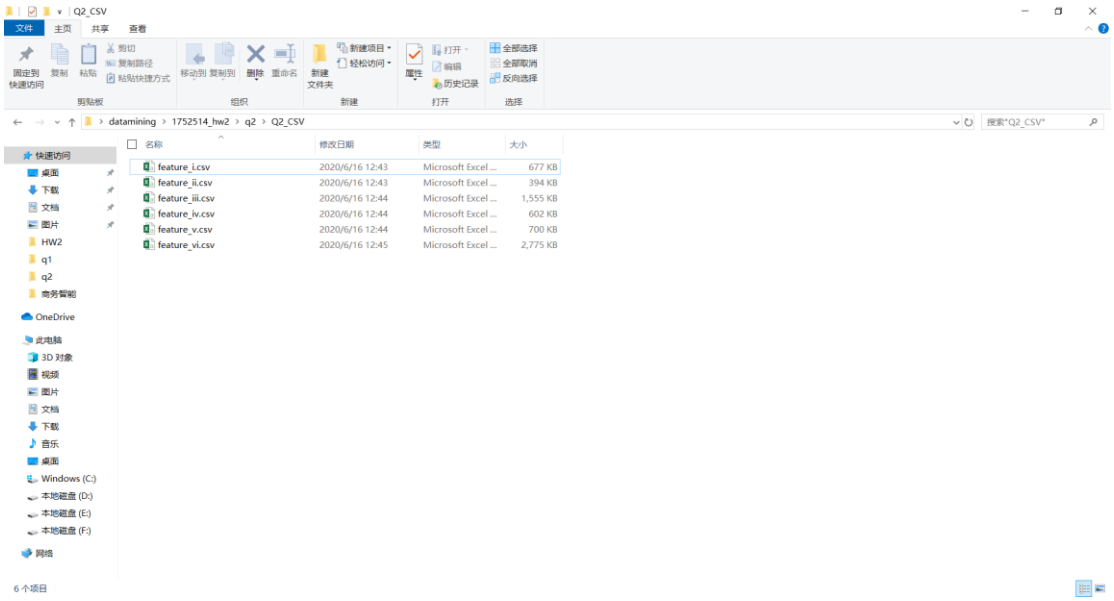
(二) 解决思路

特征工程生成的结果也是一系列 csv 文件, 根据不同设计的特征方案对数据进行进一步的处理, 因为全部数据的数据量在 20000 条左右, 这对于训练模型来说有点多也有点慢, 所以从全部数据中选择一个子集进行特征工程分析, 最后得出的数据量在 8000 条保证模型的训练量, 同时又不会让模型训练太耗时。

同时在之后的模型训练时, 训练集的标签数据这里选取商品的销量, 只可以是整数型, 而在原数据集中商品销量全都是浮点数类型, 并且很多都是大于 0 小于 1 的数据, 这样的数据在模型拟合时无法使用。所以我在特征工程中取销量精度为小数点后三位, 将所有商品的销量乘上 1000 取整。这样既可以保证正常模型训练, 最后将预测的销量除以 1000 保证至少 3 位的精度。

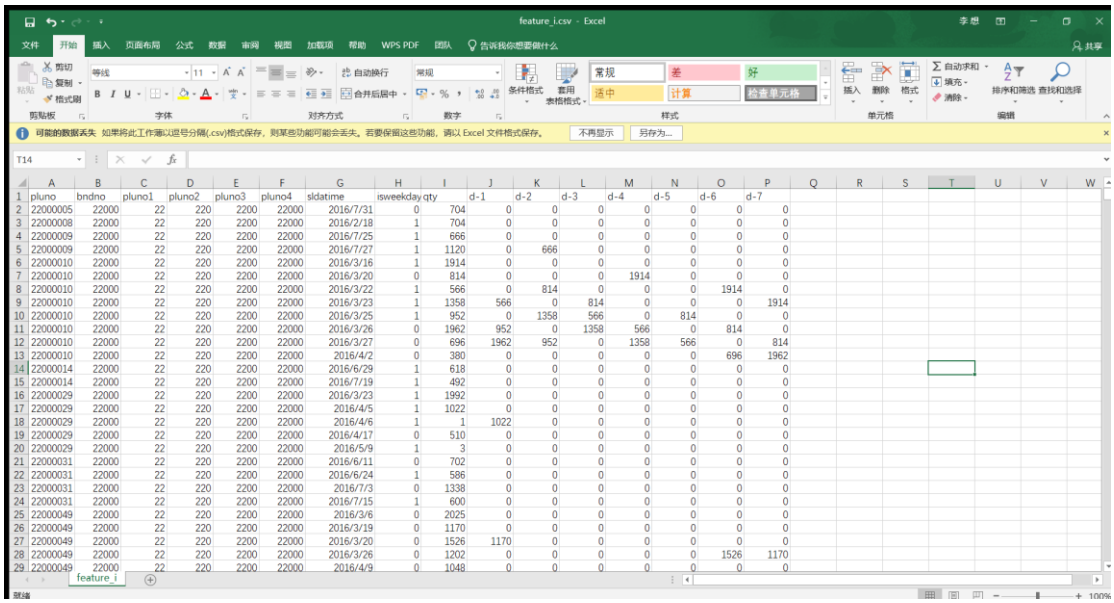
(三) 特征工程结果

下面是所有的 csv 文件的截图, 所有 csv 文件都会附在文件夹中:

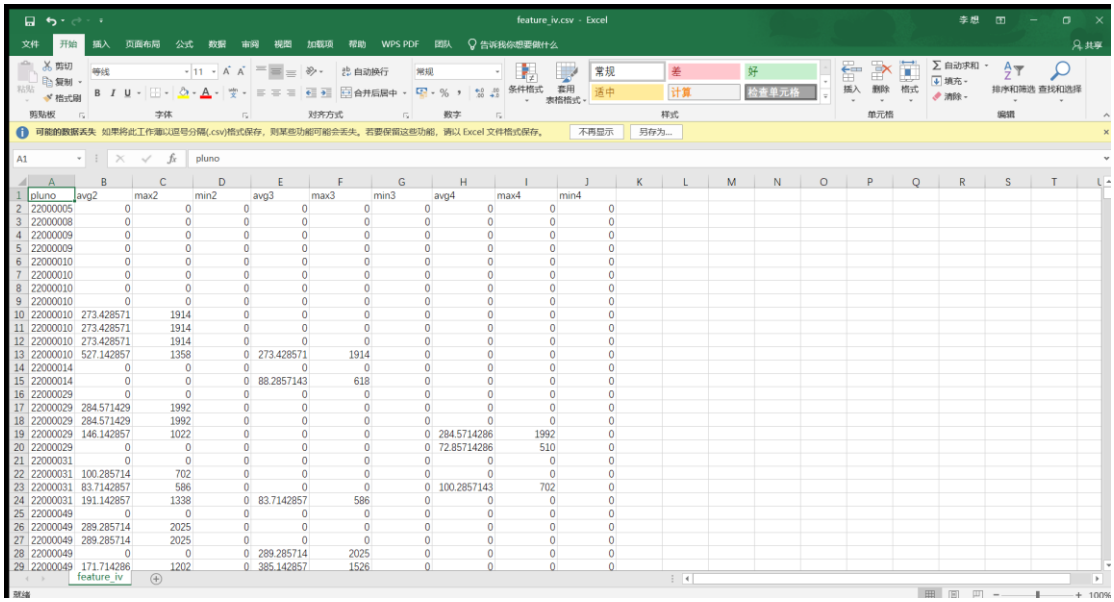


下面是截取了特征工程 i 和特征工程 iv 的部分 csv 数据，这两个特征工程代表的分别是**类别类数据**（主要表征商品所属品类、品牌）、**统计数值类数据**（主要表征销量的平均值、最大值、最小值），二者都包含了**时间类数据**（这里没有使用距离最早日期的天数表示，是想让特征工程表更加直观，但在后续的模型训练时会将时间数据转化为离散的天数，这样能更好地训练所需预测模型）：

Feature_i:



Feature_iv:



pluno	avg2	max2	min2	avg3	max3	min3	avg4	max4	min4
22000005	0	0	0	0	0	0	0	0	0
22000008	0	0	0	0	0	0	0	0	0
22000009	0	0	0	0	0	0	0	0	0
22000009	0	0	0	0	0	0	0	0	0
22000010	0	0	0	0	0	0	0	0	0
22000010	0	0	0	0	0	0	0	0	0
22000010	0	0	0	0	0	0	0	0	0
22000010	0	0	0	0	0	0	0	0	0
22000010	273.428571	1914	0	0	0	0	0	0	0
22000010	273.428571	1914	0	0	0	0	0	0	0
22000010	273.428571	1914	0	0	0	0	0	0	0
22000010	527.142857	1358	0	273.428571	1914	0	0	0	0
22000014	0	0	0	0	0	0	0	0	0
22000014	0	0	0	88.2857143	618	0	0	0	0
22000029	0	0	0	0	0	0	0	0	0
22000029	284.571429	1992	0	0	0	0	0	0	0
22000029	284.571429	1992	0	0	0	0	0	0	0
22000029	146.142857	1022	0	0	0	0	284.5714286	1992	0
22000029	0	0	0	0	0	0	72.85714286	510	0
22000031	0	0	0	0	0	0	0	0	0
22000031	100.285714	702	0	0	0	0	0	0	0
22000031	83.7142857	586	0	0	0	0	100.2857143	702	0
22000031	191.142857	1338	0	83.7142857	586	0	0	0	0
22000049	0	0	0	0	0	0	0	0	0
22000049	289.285714	2025	0	0	0	0	0	0	0
22000049	289.285714	2025	0	0	0	0	0	0	0
22000049	0	0	0	289.285714	2025	0	0	0	0
22000049	171.714286	1202	0	385.142857	1526	0	0	0	0

其他的 csv 都是按照相应的特征量设计进行处理。

(四)　详细实现

- **辅助函数**

  `is_weekday(date_str)`：判断是否是周末，是周六或周天就返回 `False`，是工作日则返回 `True`

  `last_week_list(date_str)`：生成给定的 `date_str` 前 7 天的日期

  `past_week_list(date_str)`：生成给定的 `date_str` 前第二周、前第三周和前第四周的日期

  `min_date()`：返回数据集中最早的日期

  `max_date()`：返回数据集中最晚的日期

  `get_date_list()`：获取返回从最早到最晚的所有日期数组

  `get_pluno_dict()`：获取返回以 `pluno` 为第一级 `key` 值，`date_str` 为第二级 `key` 值的字典

  `get_bndno_dict()`：获取返回以 `bndno` 为第一级 `key` 值，`date_str` 为第二级 `key` 值的字典

  `get_pluno_level_dict(lev)`：根据输入的 `lev` 即商品品类级别，获取并返回以相应品类 `pluno_lev` 为第一级 `key` 值，`date_str` 为第二级 `key` 值的字典

- **特征工程函数**

  生成 feature_i 的函数:

```
def feature_i(self):
    result = {}  # 结果数组存储最终特征工程结果
    # 设置 csv 文件头部
    head = ['pluno','bndno','pluno1','pluno2','pluno3','pluno4','sldatime'
        ,'isweekday','qty','d-1','d-2','d-3','d-4','d-5','d-6','d-7']
    # 创建文件对象
    path = "feature_i.csv"
    f = open(path, 'w', encoding='utf-8', newline='' "")
    # 基于文件对象构建 csv 写入对象
    csv_writer = csv.writer(f)
    # 构建列表头
    csv_writer.writerow(head)
    # 获取 pluno,date 二级字典用来查找相应数据
    history_dict = self.get_pluno_dict()
    # 遍历源数据
    for index,row in self.data.iterrows():
        pluno = row['pluno']
        pluno1 = int(pluno / 1000000)
        # 选取所有数据中的一个子集
        if pluno1 in pluno_key:
            # 我将 bndno 设置为第四季品类的理由是:
            # 这些没有 bndno 的商品都可以分为相应的商品类别，比如每一个苹果可能都没有 bndno
            # 但是它们都是苹果就相当于一个品牌，给每一个苹果设置成一个品牌反而会给模型造成误解
            bndno = int(pluno / 1000)
            pluno2 = int(pluno / 100000)
            pluno3 = int(pluno / 10000)
            pluno4 = int(pluno / 1000)
            sldatime = row['sldatime'][0:10]
            # 判断是否是周末
            if self.is_weekday(sldatime):
                isweekday = 1
            else:
                isweekday = 0
            qty = row['qty']
            # 获取之前一周的日期
            weeklist = self.last_week_list(sldatime)
            # 合并同一 pluno、sldatime 的商品
            if pluno in result:
```

```
                if sldatime in result[pluno]:
                    result[pluno][sldatime][8] += qty
                else:
                    result[pluno][sldatime] = [pluno, bndno, pluno1, pluno2, pluno3, pluno4, sldatime, isweekday, qty]
            else:
                result[pluno] = {}
                result[pluno][sldatime] = [pluno, bndno, pluno1, pluno2, pluno3, pluno4, sldatime, isweekday, qty]
            # 获取前一周内每天该商品的销量
            if pluno in result:
                if sldatime in result[pluno]:
                    li = result[pluno][sldatime]
                    if len(li) == 9:
                        for date in weeklist:
                            min_date = datetime.datetime.strptime('2016-02-01','%Y-%m-%d')
                            this_date = datetime.datetime.strptime(date,'%Y-%m-%d')
                            # 当日期超出最早日期时该日期销量设为0
                            if this_date < min_date:
                                li.append(0.0)
                            else:
                                li.append(history_dict[pluno][date])
    # 创建每一行数据
    for key1 in sorted(result.keys()):
        dict1 = result[key1]
        for key2 in sorted(dict1.keys()):
            rec = dict1[key2]
            csv_writer.writerow(rec)
    # 关闭文件
    f.close()
```

生成 feature_iv 的函数:

```
def feature_iv(self):
    result = {}  # 结果数组存储最终特征工程结果
    # 设置 csv 文件头部
    head = ['pluno', 'avg2', 'max2', 'min2', 'avg3', 'max3', 'min3', 'avg4',

            'max4', 'min4']
    # 创建文件对象
    path = "feature_iv.csv"
    f = open(path, 'w', encoding='utf-8', newline='' "")
```

```
    # 基于文件对象构建 csv 写入对象
    csv_writer = csv.writer(f)
    # 构建列表头
    csv_writer.writerow(head)
    # 获取 pluno,date 二级字典用来查找相应数据
    history_dict = self.get_pluno_dict()
    # 遍历源数据
    for index, row in self.data.iterrows():
        pluno = row['pluno']
        pluno1 = int(pluno / 1000000)
        sldatime = row['sldatime'][0:10]
        if pluno1 in pluno_key:
            weeklist = self.past_week_list(sldatime)
            # 合并同一 pluno、sldatime 的商品
            if pluno in result:
                if sldatime in result[pluno]:
                    continue
                else:
                    result[pluno][sldatime] = []
                    result[pluno][sldatime].append(pluno)
            else:
                result[pluno] = {}
                result[pluno][sldatime] = []
                result[pluno][sldatime].append(pluno)
            # 添加数据
            if pluno in result:
                if sldatime in result[pluno]:
                    li = result[pluno][sldatime]
                    if len(li) == 1:
                        avg = 0.0
                        max = 0.0
                        min = float('inf')
                        # 循环遍历前 2、3、4 周，获取每周的 avg、max、min
                        for week in weeklist:
                            # 遍历一周中的每天
                            for date in week:
                                min_date = datetime.datetime.strptime('2016-02-01', '%Y-%m-%d')
                                this_date = datetime.datetime.strptime(date, '%Y-%m-%d')
                                # 因为早于最早日期的销量全是 0,所以 avg 不需要加
                                if this_date > min_date:
                                    avg += history_dict[pluno][date]
                                    if history_dict[pluno][date] > max:
```

```
                                        max = history_dict[pluno][date]
                                    if history_dict[pluno][date] < min:
                                        min = history_dict[pluno][date]
                                else:
                                    min = 0.0
                            avg = avg / 7
                            li.append(avg)
                            li.append(max)
                            li.append(min)
                            # 重置 avg、max、min
                            avg = 0.0
                            max = 0.0
                            min = float('inf')
    # 创建每一行数据
    for key1 in sorted(result.keys()):
        dict1 = result[key1]
        for key2 in sorted(dict1.keys()):
            rec = dict1[key2]
            csv_writer.writerow(rec)
    # 关闭文件
    f.close()
```

(五)　遇到的问题

我总共更改了三次代码，第一次的问题是使用字典存储时使用 pluno 编号作为 key 值，导致最后拥有同一个 pluno 编号的商品记录被最后一个覆盖，数据丢失严重。

第二次的问题是我改变存储结构，使用数组将每一个商品记录存储下来，这在第二第三问都没有问题。但到了 RSE 分析时，因为我的真实值是从 pluno、date_str 为二级键值的字典中获取，它将同一 pluno、sldatime 的数据合并。而预测值是从第三问的结果 csv 文件中读取，每一个 pluno、sldatime 自成一条元组，导致同一商品如果在一天内卖出多次时，真实值大于预测值，最终 RSE 结果明显有问题。

最后我更改存储结构，采用 pluno、sldatime 二级键值的字典存储，解决了数据丢失和真实值与预测值逻辑上不匹配的问题，最后得到正常范围内的 RSE。