Q3 分析报告

(一) 问题描述

未来销量预测：针对训练数据中商品每天的当日销量为目标特征、其他特征（即历史信息）均为 属性特征，利用 **SVM**、**决策树**、**随机森林**、**MLP** 等四个方法进行建模，预测测试数据中某商品 对应日期当日（标记为 d'）至第 6 日（d'+6）共计 7 天的每日销量，可考虑如下算法：首先完成 商品 d'当日的销量预测，然后利用该预测销量更新上述 b)的相关特征，继续预测 d'+1 当日销量。。。重复该步骤，直至完成第 6 日（d'+6）当日销量预测。

(二) 解决思路

预测结果存储为一系列 csv 文件，使用不同的特征工程数据，将其划分为训练集、测试集，使用训练集对模型进行训练，然后用训练好的模型对测试集进行回归预测。

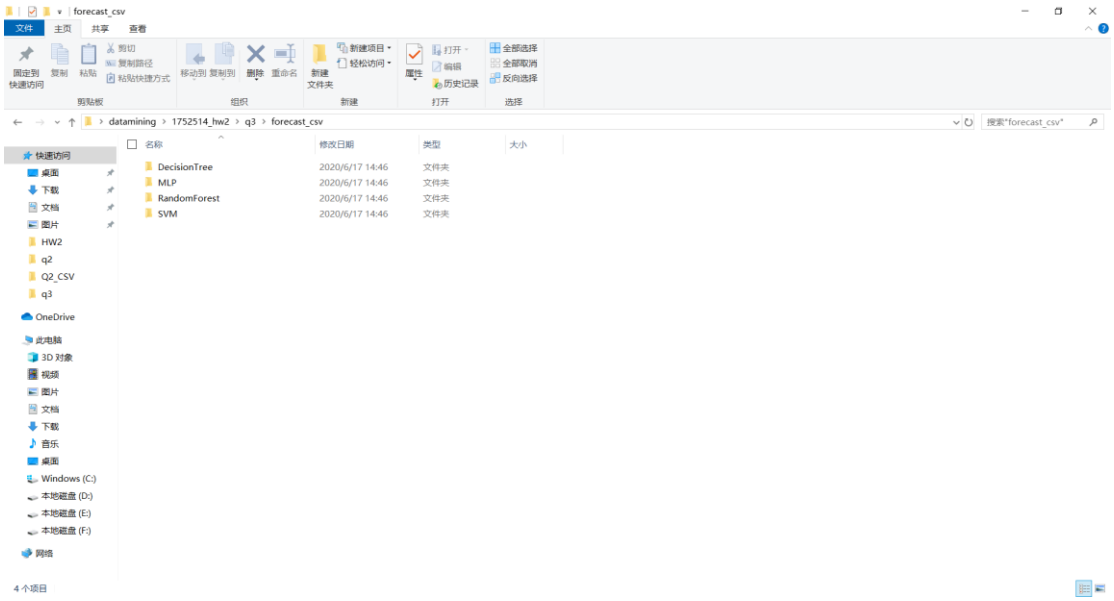
对于不同的特征工程，划分训练集和测试集的具体方法不同。对于特征工程 i~iii 来说需要舍弃数据中日期在 2016-02-01 到 2016-02-07、2016-07-25 到 2016-07-31 的闭区间之内的数据，这部分数据无法预测 d~d+6 天的销量，然后将剩下的数据根据时间序列前 80%划分为训练集，后 20%划分为测试集即可。

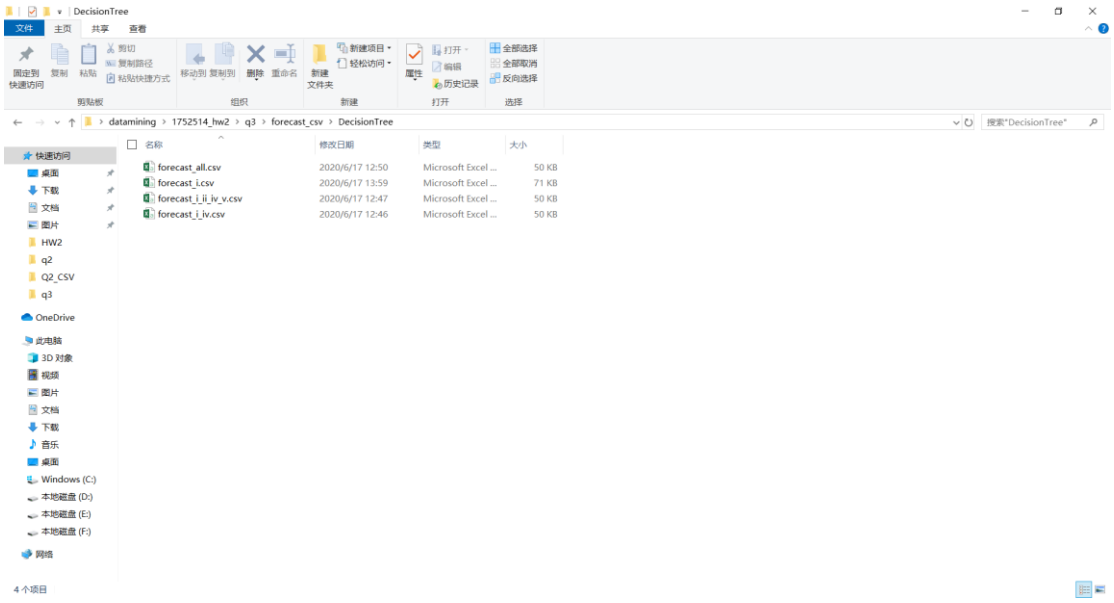
对于特征工程 iv~vi 来说，不仅需要舍弃数据中日期在 2016-02-01 到 2016-02-07、2016-07-25 到 2016-07-31 的闭区间之内的数据，还需要舍弃 2016-02-08 到 2016-02-29 闭区间内的数据，因为位于该区间内的数据没有 d-8 到 d-28 的完整特征数据。

在每次预测结束后，先将预测结果放入结果数组中，然后将预测结果逐条加入训练集中，最后更新测试集，进行后一天的预测。

(三) 预测结果

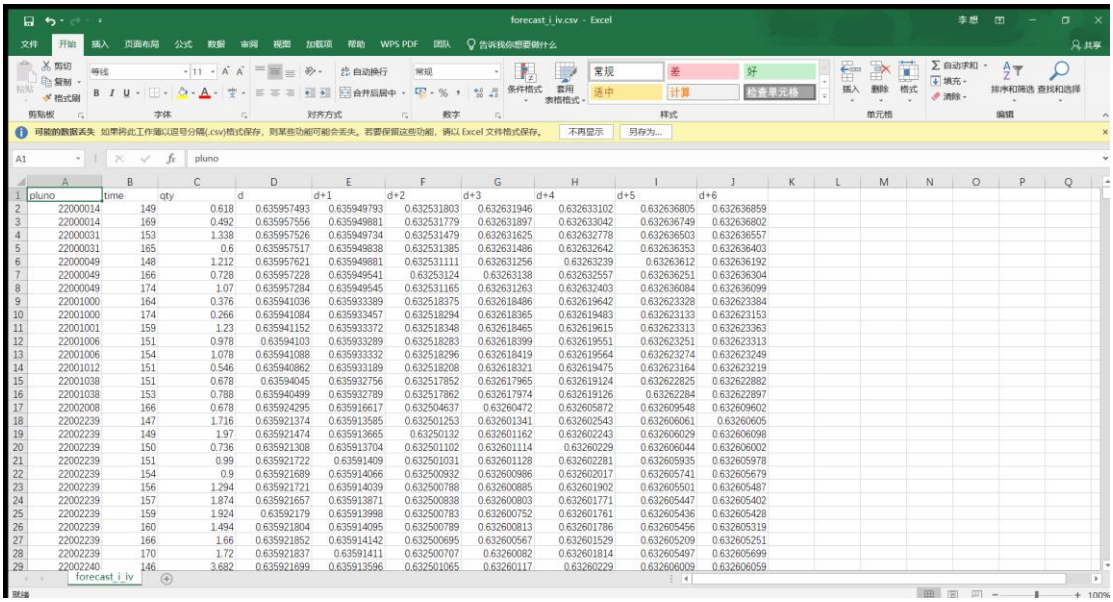
下面是所有的 csv 文件的截图，所有 csv 文件都会附在文件夹中：





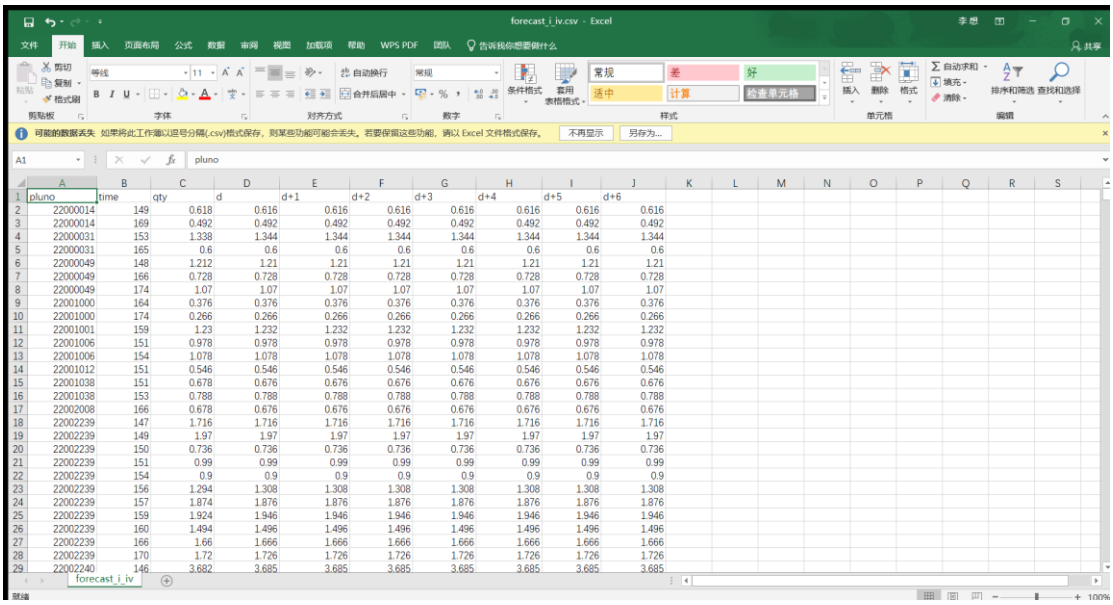
下面是截取了使用特征工程 i、iv 的数据，利用 SVM、决策树、随机森林、MLP 四种不同预测模型预测得到的未来 7 天的销量数据:

SVM:



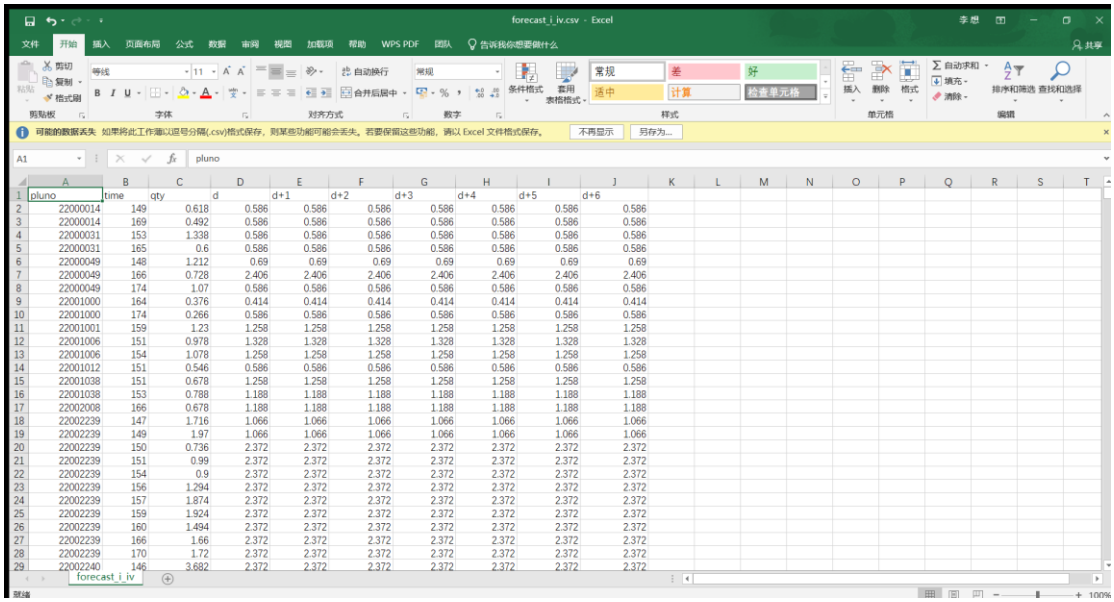
pluno	time	qty	d	d+1	d+2	d+3	d+4	d+5	d+6
22000014	149	0.618	0.635957493	0.635949793	0.632531803	0.632631946	0.632633102	0.632636805	0.632636859
22000014	169	0.492	0.635957556	0.635949881	0.632531779	0.632631897	0.632633042	0.632636749	0.632636802
22000031	153	1.338	0.635957526	0.635949734	0.632531479	0.632631625	0.632632778	0.632636503	0.632636557
22000031	165	0.6	0.635957517	0.635949838	0.632531385	0.632631486	0.632632642	0.632636353	0.632636403
22000049	148	1.212	0.635957621	0.635949881	0.632531111	0.632631256	0.63263239	0.63263612	0.632636192
22000049	166	0.728	0.635957228	0.635949541	0.63253124	0.63263138	0.632632557	0.632636251	0.632636304
22000049	174	1.07	0.635957284	0.635949545	0.632531165	0.632631263	0.632632403	0.632636084	0.632636099
22001000	164	0.376	0.635941036	0.635933389	0.632518375	0.632618486	0.632619642	0.632623328	0.632623384
22001000	174	0.266	0.635941084	0.635933457	0.632518294	0.632618365	0.632619483	0.632623133	0.632623153
22001001	159	1.23	0.635941152	0.635933372	0.632518348	0.632618465	0.632619615	0.632623313	0.632623363
22001006	151	0.978	0.63594103	0.635933289	0.632518283	0.632618399	0.632619551	0.632623251	0.632623313
22001006	154	1.078	0.635941088	0.635933332	0.632518296	0.632618419	0.632619564	0.632623274	0.632623249
22001012	151	0.546	0.635940862	0.635933189	0.632518208	0.632618321	0.632619475	0.632623164	0.632623219
22001038	151	0.678	0.63594045	0.635932756	0.632517852	0.632617965	0.632619124	0.632622825	0.632622882
22001038	153	0.788	0.635940499	0.635932789	0.632517862	0.632617974	0.632619126	0.63262284	0.632622897
22002008	166	0.678	0.635924295	0.635916617	0.632504637	0.63260472	0.632605872	0.632609548	0.632609602
22002239	147	1.716	0.635921374	0.635913585	0.632501253	0.632601341	0.632602543	0.632606061	0.63260605
22002239	149	1.97	0.635921474	0.635913665	0.63250132	0.632601162	0.632602243	0.632606029	0.632606098
22002239	150	0.736	0.635921308	0.635913704	0.632501102	0.632601114	0.63260229	0.632606044	0.632606002
22002239	151	0.99	0.635921722	0.63591409	0.632501031	0.632601128	0.632602281	0.632605935	0.632605978
22002239	154	0.9	0.635921689	0.635914066	0.632500932	0.632600986	0.632602017	0.632605741	0.632605679
22002239	156	1.294	0.635921721	0.635914039	0.632500788	0.632600885	0.632601902	0.632605501	0.632605487
22002239	157	1.874	0.635921657	0.635913871	0.632500838	0.632600803	0.632601771	0.632605447	0.632605402
22002239	159	1.924	0.63592179	0.635913998	0.632500783	0.632600752	0.632601761	0.632605436	0.632605428
22002239	160	1.494	0.635921804	0.635914095	0.632500789	0.632600813	0.632601786	0.632605456	0.632605319
22002239	166	1.66	0.635921852	0.635914142	0.632500695	0.632600567	0.632601529	0.632605209	0.632605251
22002239	170	1.72	0.635921837	0.63591411	0.632500707	0.63260082	0.632601814	0.632605497	0.632605699
22002240	146	3.682	0.635921699	0.635913596	0.632501065	0.63260117	0.63260229	0.632606009	0.632606059

DecisionTree:



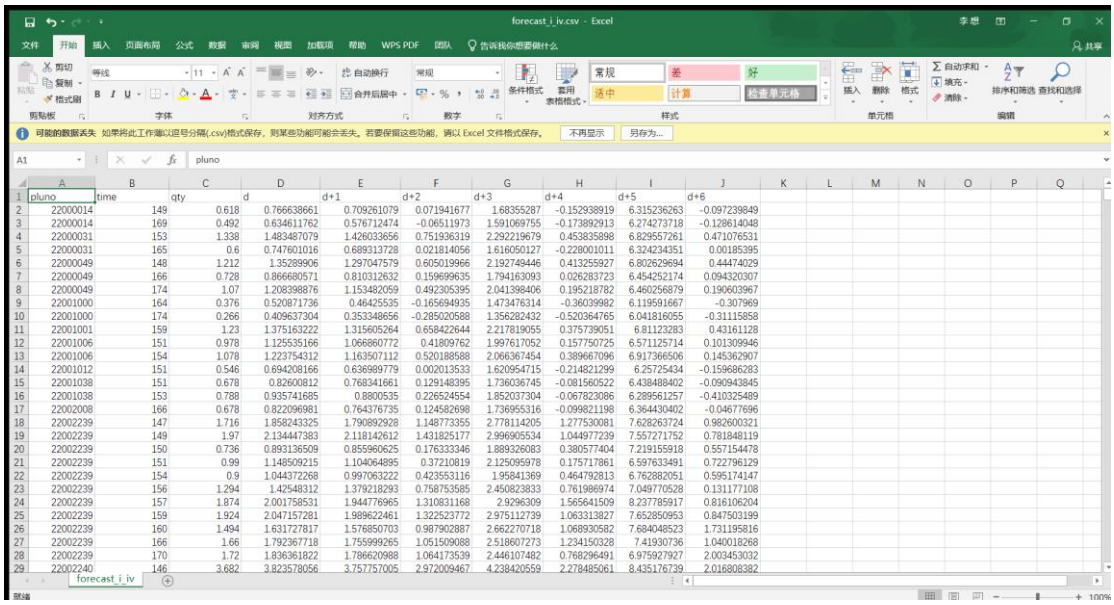
pluno	time	qty	d	d+1	d+2	d+3	d+4	d+5	d+6
22000014	149	0.618	0.616	0.616	0.616	0.616	0.616	0.616	0.616
22000014	169	0.492	0.492	0.492	0.492	0.492	0.492	0.492	0.492
22000031	153	1.338	1.344	1.344	1.344	1.344	1.344	1.344	1.344
22000031	165	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
22000049	148	1.212	1.21	1.21	1.21	1.21	1.21	1.21	1.21
22000049	166	0.728	0.728	0.728	0.728	0.728	0.728	0.728	0.728
22000049	174	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07
22001000	164	0.376	0.376	0.376	0.376	0.376	0.376	0.376	0.376
22001000	174	0.266	0.266	0.266	0.266	0.266	0.266	0.266	0.266
22001001	159	1.23	1.232	1.232	1.232	1.232	1.232	1.232	1.232
22001006	151	0.978	0.978	0.978	0.978	0.978	0.978	0.978	0.978
22001006	154	1.078	1.078	1.078	1.078	1.078	1.078	1.078	1.078
22001012	151	0.546	0.546	0.546	0.546	0.546	0.546	0.546	0.546
22001038	151	0.678	0.676	0.676	0.676	0.676	0.676	0.676	0.676
22001038	153	0.788	0.788	0.788	0.788	0.788	0.788	0.788	0.788
22002008	166	0.678	0.676	0.676	0.676	0.676	0.676	0.676	0.676
22002239	147	1.716	1.716	1.716	1.716	1.716	1.716	1.716	1.716
22002239	149	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97
22002239	150	0.736	0.736	0.736	0.736	0.736	0.736	0.736	0.736
22002239	151	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
22002239	154	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
22002239	156	1.294	1.308	1.308	1.308	1.308	1.308	1.308	1.308
22002239	157	1.874	1.876	1.876	1.876	1.876	1.876	1.876	1.876
22002239	159	1.924	1.946	1.946	1.946	1.946	1.946	1.946	1.946
22002239	160	1.494	1.496	1.496	1.496	1.496	1.496	1.496	1.496
22002239	166	1.66	1.666	1.666	1.666	1.666	1.666	1.666	1.666
22002239	170	1.72	1.726	1.726	1.726	1.726	1.726	1.726	1.726
22002240	146	3.682	3.685	3.685	3.685	3.685	3.685	3.685	3.685

RandomForest:



pluno	time	qty	d	d+1	d+2	d+3	d+4	d+5	d+6
22000014	149	0.618	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22000014	169	0.492	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22000031	153	1.338	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22000031	165	0.6	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22000049	148	1.212	0.69	0.69	0.69	0.69	0.69	0.69	0.69
22000049	166	0.728	2.406	2.406	2.406	2.406	2.406	2.406	2.406
22000049	174	1.07	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22001000	164	0.376	0.414	0.414	0.414	0.414	0.414	0.414	0.414
22001000	174	0.266	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22001001	159	1.23	1.258	1.258	1.258	1.258	1.258	1.258	1.258
22001006	151	0.978	1.328	1.328	1.328	1.328	1.328	1.328	1.328
22001006	154	1.078	1.258	1.258	1.258	1.258	1.258	1.258	1.258
22001012	151	0.546	0.586	0.586	0.586	0.586	0.586	0.586	0.586
22001038	151	0.678	1.258	1.258	1.258	1.258	1.258	1.258	1.258
22001038	153	0.788	1.188	1.188	1.188	1.188	1.188	1.188	1.188
22002008	166	0.678	1.188	1.188	1.188	1.188	1.188	1.188	1.188
22002239	147	1.716	1.066	1.066	1.066	1.066	1.066	1.066	1.066
22002239	149	1.97	1.066	1.066	1.066	1.066	1.066	1.066	1.066
22002239	150	0.736	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	151	0.99	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	154	0.9	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	156	1.294	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	157	1.874	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	159	1.924	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	160	1.494	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	166	1.66	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002239	170	1.72	2.372	2.372	2.372	2.372	2.372	2.372	2.372
22002240	146	3.682	2.372	2.372	2.372	2.372	2.372	2.372	2.372

MLP:



pluno	time	qty	d	d+1	d+2	d+3	d+4	d+5	d+6
22000014	149	0.618	0.766638661	0.709261079	0.071941677	1.68355287	-0.152938919	6.315236263	-0.097239849
22000014	169	0.492	0.634611762	0.576712474	-0.06511973	1.591069755	-0.173892913	6.274273718	-0.128614048
22000031	153	1.338	1.483487079	1.426033656	0.751936319	2.292219679	0.453835898	6.829557261	0.471076531
22000031	165	0.6	0.747601016	0.689313728	0.021814056	1.616050127	-0.228001011	6.324234351	0.00185395
22000049	148	1.212	1.35289906	1.297047579	0.605019966	2.192749446	0.413255927	6.802629694	0.44474029
22000049	166	0.728	0.866680571	0.810312632	0.159699635	1.794163093	0.026283723	6.454252174	0.094320307
22000049	174	1.07	1.208398876	1.153482059	0.492305395	2.041398406	0.195218782	6.460256879	0.190603967
22001000	164	0.376	0.520871736	0.46425535	-0.165694935	1.473476314	-0.36039982	6.119591667	-0.307969
22001000	174	0.266	0.409637304	0.353348656	-0.285020588	1.356282432	-0.520364765	6.041816055	-0.31115858
22001001	159	1.23	1.375163222	1.315605264	0.658422644	2.217819055	0.375739051	6.81123283	0.43161128
22001006	151	0.978	1.125535166	1.066860772	0.41809762	1.997617052	0.157750725	6.571125714	0.101309946
22001006	154	1.078	1.223754312	1.163507112	0.520188588	2.066367454	0.389667096	6.917366506	0.145362907
22001012	151	0.546	0.694208166	0.636989779	0.002013533	1.620954715	-0.214821299	6.25725434	-0.159686283
22001038	151	0.678	0.82600812	0.768341661	0.129148395	1.736036745	-0.081560522	6.438488402	-0.090943845
22001038	153	0.788	0.935741685	0.8800535	0.226524554	1.852037304	-0.067823086	6.289561257	-0.410325489
22002008	166	0.678	0.822096981	0.764376735	0.124582698	1.736955316	-0.099821198	6.364430402	-0.04677696
22002239	147	1.716	1.858243325	1.790892928	1.148773355	2.778114205	1.277530081	7.628263724	0.982600321
22002239	149	1.97	2.134447383	2.118142612	1.431825177	2.996905534	1.044977239	7.557271752	0.781848119
22002239	150	0.736	0.893136509	0.855960625	0.176333346	1.889326083	0.380577404	7.219155918	0.557154478
22002239	151	0.99	1.148509215	1.104064895	0.37210819	2.125095978	0.175717861	6.597633491	0.722796129
22002239	154	0.9	1.044372268	0.997063222	0.423553116	1.95841369	0.464792813	6.762882051	0.595174147
22002239	156	1.294	1.42548312	1.379218293	0.758753585	2.450823833	0.761986974	7.049770528	0.131177108
22002239	157	1.874	2.001758531	1.944776965	1.310831168	2.9296309	1.565641509	8.237785917	0.816106204
22002239	159	1.924	2.047157281	1.989622461	1.322523772	2.975112739	1.063313827	7.652850953	0.847503199
22002239	160	1.494	1.631727817	1.576850703	0.987902887	2.662270718	1.068930582	7.684048523	1.731195816
22002239	166	1.66	1.792367718	1.755999265	1.051509088	2.518607273	1.234150328	7.41930736	1.040018268
22002239	170	1.72	1.836361822	1.786620988	1.064173539	2.446107482	0.768296491	6.975927927	2.003453032
22002240	146	3.682	3.823578056	3.757757005	2.972009467	4.238420559	2.278485061	8.435176739	2.016808382

可以看出不同的模型预测的结果不相同，前三种从预测结果看对日期特征不敏感，表现在虽然当天的预测销量与真实的接近，但未来几天的销量没有什么变化；而 MLP 模型预测结果似乎对时间更加敏感，它的当天预测销量与随机森林、决策树相比浮动更大，但未来几天的预测销量有涨有落。

(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

- **预测函数**

这里展示使用 feature_i 和 feature_iv 特征工程的预测函数作为代表：

```
def forecast_i_iv():
    # 划分训练集和测试集的时间节点
    training_end = datetime.datetime.strptime('2016-06-25', '%Y-%m-%d')
    training_start = datetime.datetime.strptime('2016-02-29', '%Y-%m-%d')
    test_end = datetime.datetime.strptime('2016-07-25', '%Y-%m-%d')
    start = datetime.datetime.strptime('2016-02-01', '%Y-%m-%d')
    all_data = []  # 存储合并之后的特征工程
    training_x = []
    training_y = []
    test_x = []
    test_y = []
    # 首先合并特征工程
    for i, row in feature_i.iterrows():
        row['sldatime'] = (datetime.datetime.strptime(row['sldatime'], '%Y-%m-%d') - start).days
        all_data.append([])
        j = 0
        while j < len(row):
```

```
            all_data[i].append(row[j])
            j += 1
    for i, row in feature_iv.iterrows():
        j = 0
        while j < len(row):
            all_data[i].append(row[j])
            j += 1
    # 划分训练集和测试集
    for row in all_data:
        time = start + datetime.timedelta(days=row[6])
        if training_start < time < training_end:
            training_x.append(row)
            training_y.append(row[8])
        elif training_end < time < test_end:
            test_x.append(row)
            test_y.append(row[8])
    # 初始化 result
    result = []
    id = 0
    while id < len(test_x):
        result.append([])
        result[id].append(test_x[id][0])
        result[id].append(test_x[id][6])
        result[id].append(test_x[id][8] / 1000)
        id += 1
    history_dict = get_pluno_dict()
    # 重复预测 7 次
    day = 0
    while day < 7:
        # 预测
        # clf = RandomForestClassifier()
        # linear,poly,rbf
        # clf = SVR(kernel="poly")
        # clf = tree.DecisionTreeClassifier(criterion='entropy')
        clf = MLPRegressor()
        clf.fit(training_x, training_y)
        predict_y = clf.predict(test_x)
        i = 0
        while i < len(test_x):
            # 保存到结果字典 result 中
            pluno = test_x[i][0]
            date_str = datetime.datetime.strftime(start + datetime.timedelta
(days=test_x[i][6]), '%Y-%m-%d')
            test_x[i][8] = predict_y[i]
```

```
61.             result[i].append(predict_y[i] / 1000)
62.             # 当预测的是当天时，其他特征量不用更新，直接添加到训练集中即可
63.             if day == 0:
64.                 training_x.append(test_x[i])
65.                 training_y.append(test_x[i][8])
66.             if day > 0:
67.                 # 更新时间序列字典
68.                 history_dict[pluno][date_str] += predict_y[i]
69.                 rec = test_x[i]
70.                 # 更新特征量 d-1/d-7
71.                 j = 0
72.                 lastweek = last_week_list(date_str)
73.                 for date in lastweek:
74.                     min_date = datetime.datetime.strptime('2016-02-01', '%Y-
    %m-%d')
75.                     this_date = datetime.datetime.strptime(date, '%Y-%m-%d')

76.                     if this_date > min_date:
77.                         rec[9 + j] = history_dict[pluno][date]
78.                     else:
79.                         rec[9 + j] = 0.0
80.                     j += 1
81.                 # 更新 avg、max、min
82.                 week_list = past_week_list(date_str)
83.                 avg = 0.0
84.                 max = 0.0
85.                 min = float('inf')
86.                 week_index = 0
87.                 # 遍历前 2、3、4 周
88.                 for week in week_list:
89.                     # 遍历一周中的每一天
90.                     for date in week:
91.                         min_date = datetime.datetime.strptime('2016-02-01',
    '%Y-%m-%d')
92.                         this_date = datetime.datetime.strptime(date, '%Y-%m-
    %d')
93.                         if this_date > min_date:
94.                             avg += history_dict[pluno][date]
95.                             if history_dict[pluno][date] > max:
96.                                 max = history_dict[pluno][date]
97.                             if history_dict[pluno][date] < min:
98.                                 min = history_dict[pluno][date]
99.                         else:
100.                             min = 0.0
```

```
                    avg = avg / 7
                    rec[16 + 3 * week_index] = avg
                    rec[17 + 3 * week_index] = max
                    rec[18 + 3 * week_index] = min
                    avg = 0.0
                    max = 0.0
                    min = float('inf')
                    week_index += 1
                #     更新所有特征量添加到训练集中
                training_x.append(rec)
                training_y.append(rec[8])
            i += 1
        # 更新日期进行下次预测
        for row in test_x:
            row[6] += 1
            date = datetime.datetime.strftime(start + datetime.timedelta(da
ys=row[6]), '%Y-%m-%d')
            if is_weekday(date):
                row[7] = 1
            else:
                row[7] = 0
        day += 1
    # 将预测结果写入 csv
    head = ['pluno', 'time', 'qty', 'd', 'd+1', 'd+2', 'd+3', 'd+4', 'd+5',
 'd+6']
    # 创建文件对象
    path = "DecisionTree_forecast_i_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)
    # 创建每一行数据
    for row in result:
        csv_writer.writerow(row)
    # 关闭文件
    f.close()
```