

## H2 数据挖掘第二次作业 q3 报告

### H3 题目要求

**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、随机森林、MLP 等 3 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。

**性能评测：**

1. 在 a) 的每日时间序列数据中，对每个商品按照安排时间从早到晚的顺序排列，分别选取该商品 80%和 20%的时序数据作为训练和测试数据
2. 对比①仅使用  $b.i$  特征、②仅使用  $b.i+b.iv$  特征、③仅使用  $b.i+b.ii+b.iii+b.iv$  特征、④使用  $b.i+b.ii+b.iii+b.iv+b.v+b.vi$  特征等 4 类场景的性能对比，并加以讨论。iii. 指标：root relative squared error (RSE)，见参考文献的公式(5)。

### H3 代码设计

### H4 数据预处理

一开始放入数据后,考虑对pluno进行处理,第一反应是改成 one-hot 编码,后来发现维度过高,于是想到了 one-hot + pca 的组合,经过试验发现容易过拟合,尤其是在时序数据随着天数的推移,测试集的分布会与原先训练集相差很大。经过试验,对随机森林使用 PCA,其他不使用, 品类结构转成了目标量 qty 的平均,通过限制小数点后位数来增强泛化能力。

```
1  # 数据预处理
2  def read_dataset(path):
3      dataset = pd.read_csv(path)
4      dataset['purchase_date']=[x.replace('/', '-') for x in
dataset["purchase_date"]]
5      dataset['purchase_date']=pd.to_datetime(dataset['purchase_date'])
6      dataset['purchase_date']=[datetime.datetime.strptime(x, '%Y-%m-%d') for
x in dataset['purchase_date']]
7      dataset['pluno'] = dataset['pluno'].astype('str')
8
9      plunos = np.unique(dataset['pluno'])
10     dates = np.unique(dataset['purchase_date'])
11     train_dates = dates
12     test_dates = dates[146:]
13
14     for i in range(1,5):
15         col = "pl_"+str(i)
16         record =
dict(dataset.loc[dataset["purchase_date"].isin(train_dates)].groupby(by=
[col]))['qty'].mean())
17         dataset[col]=[round(record[x],2) for x in dataset[col]]
18     '''
19     one-hot + pca 在Radom Forest时添加
20     #           # one-hot
```

```

21 #         enc=OneHotEncoder(categories='auto')
22 #         enc.fit(np.array(dataset[col]).reshape(-1,1))
23 #         dplunos =
enc.transform(np.array(dataset[col]).reshape(-1,1)).toarray()
24
25 #         # pca
26 #         pca=PCA(n_components=i)
27 #         pca.fit(dplunos)
28 #         compressed_dplunos = pca.transform(dplunos)
29 #         compressed_dplunos = np.around(compressed_dplunos, decimals=2)
30
31 #         dataset[col] =compressed_dplunos[:,0]
32 #         for c in range(1,compressed_dplunos.shape[1]):
33 #             new_pl = col+'_'+str(c)
34 #             col_name = dataset.columns.tolist()
35 #             col_name.insert(col_name.index(col)+1,new_pl)
36 #             dataset=dataset.reindex(columns=col_name)
37 #             dataset[new_pl] = compressed_dplunos[:,c]
38 '''

```

#### H4 更新函数及预测

计算 $RSE$ 的时候并不是先全部算完,再比较 $RSE$ ,而是直接每预测一个样本的 $RSE$ 值.

```

1  '''
2  @description: 计算测试集RSE的分子
3  @params:
4      - test_set: 测试集
5      - regressor: 指定回归模型,分别是mlp, random forest 和 svm
6      - days: 向前预测的步数(天数)
7  @output: RSE分子
8  '''
9  def calculate_rse(test_set,regressor,days):
10     total_rse,base_rse = 0,0
11     for pluno in plunos:
12         #表示7天
13         test_data = copy.deepcopy(test_set[pluno]['data'])
14         test_target = test_set[pluno]['target']
15         for index in range(test_data.shape[0]-7):
16             total_rse =
total_rse+sample_rse(test_data,test_target,index,days,regressor)
17     return total_rse
18 '''
19 @description: 计算测试集RSE的分母
20 @params:
21     - test_set: 测试集
22     - day: 向前预测的步数(天数)
23 @output: RSE分母
24 '''

```

```

25 def get_base_rse(test_set, day):
26     total_pluno_qty = 0
27     # 遍历每个商品
28     for pluno in test_set.keys():
29         total_pluno_qty = total_pluno_qty + sum_pluno_qty(test_set[pluno]
30         ['target'], day)
31
32     # 算出分母里的平均
33     avg_pluno_qty = total_pluno_qty / (len(test_set)*(len(test_dates)-7))
34
35     total_base_rse = 0
36     for pluno in test_set.keys():
37         total_base_rse = total_base_rse + sum_pluno_qty((test_set[pluno]
38         ['target'] - avg_pluno_qty)**2, day)
39     return total_base_rse
40
41 '''
42 @description: 计算测试集RSE的分母中的平均值
43 @params:
44     - test_set: 测试集
45     - day: 向前预测的步数(天数)
46 @output: RSE分母中的平均值
47 '''
48
49 def sum_pluno_qty(targets, day):
50     sum_qty = 0
51     length = targets.shape[0]
52     return np.sum(targets[:length-7])
53
54 # for d in range(day):
55 #     sum_qty = sum_qty + np.sum(targets[day:length-6+day])

```

#### H4 调参原则

##### 1. MLP:

通过采用尽可能简单的结构以及增加alpha保证模型的泛化能力, batch\_size 小一些, 使得模型能够落在平缓的较优点上.

##### 2. Random Forest:

增加树的数目来增加预测的准确性, 减低树的深度来增强泛化能力.

##### 3. SVM:

SVM 测出来始终在0.9-1.1之间. 无论怎么调参和特征工程都没有用

#### H3 实验结果

#### H4 四种场景下三种机器学习模型比较

##### 1. 仅使用 b.i 特征



---

2. 仅使用 b.i+b.iv 特征

---



3. 仅使用 b.i+b.ii+b.iii+b.iv 特征



4. 使用 b.i+b.ii+b.iii+b.iv+b.v+b.vi



H4 四种场景下三种机器单步预测比较

	Scene 1	Scene 2	Scene 3	Scene 4
Forest	0.408	0.366	0.366	0.221
MLP	0.875	0.694	0.756	0.923
SVM	0.999	0.932	0.972	1.293

H3 实验结果分析

- 1. 可以看出,三种模型在时序数据的预测上面的表现非常不一致。
  - 森林在第一天表现较好,后面表现很差,且时序特征越多,前几天表现越好,最后几天的表现越差。
  - MLP的表现则是随着天数的增加缓慢的变差,总体表现最好
  - SVM的表现则非常的稳定,即对时间不敏感,也无法拟合,始终给出一个接近均值的答案。
- 2. 我认为这反应了时间序列预测问题的两重性。假设原先训练集和测试集有接近的分布,随着时间的推移,训练集和测试集的分布会因为时序数据的更新而不在同一种分布上。换句话说,学习能力强的模型,在后几天的预测中反而表现会更差。如果希望在全周期内模型都有不错的表现,我们一方面要增强模型的学习能力,又需要增强模型的泛化能力。

为此,我做出了以下努力:

- 对品类数据编码以增强数据,将四级品类结构分组统计 qty 的平均值.

如图所示,对 22000005 来说,其 pl\_1 值为所有开头为 22 的商品的 qty 的平均值.

	pluno	purchase_date	bndno	pl_1	pl_2	pl_3	pl_4	is_workday	qty	prev_d1	...	pl3_4week_min	pl4_2week_avg	pl4_2week_max	pl4_2week_mi
0	22000005	2016-07-31	0	0.06	0.06	0.06	0.04	0	0.704	0.0	...	0.0	0.027893	1.070	0.
1	22000008	2016-02-18	0	0.06	0.06	0.06	0.04	1	0.704	0.0	...	0.0	0.000000	0.000	0.
2	22000009	2016-07-25	0	0.06	0.06	0.06	0.04	1	0.666	0.0	...	0.0	0.023714	0.728	0.
3	22000009	2016-07-27	0	0.06	0.06	0.06	0.04	1	1.120	0.0	...	0.0	0.032500	0.728	0.
4	22000010	2016-03-16	0	0.06	0.06	0.06	0.04	1	1.914	0.0	...	0.0	0.036179	2.026	0.

5 rows × 105 columns

- 对于随机森林,寻找合适的选取树的数目和深度,同时发现在树模型中,品类结构对于 qty 的直接表示会导致模型在后几天的 RSE 指标爆炸,所以对其进行了主成分分析,舍弃比重不大的成分. 时模型在第7天的 RSE 表现从40以上降低到4-5.
  - 对于MLP,由于神经网络对于品类结构的权重能够自主的学习,因此主成分分析用处不大.在模型的结构上,除最后一种情况下,尽可能选取两层隐藏层的结构,同时核数尽可能小,L2系数与之匹配.
  - 对于SVM,调低 C.
- 3. 从四个场景的表现来看,场景2的数据对于预测比较有用。
  - 4. 通过实验数据比较,除了在第一天随机森林有较大优势,但长久来看,MLP是更好的选择.考虑到用于时间序列分析的诸如LSTM等都是基于神经网络的,我认为这个结果是正常的。