Rossmann 药妆店分类报告

机器学习纳米学位-毕业项目

宋振
2018 年 11 月 26 日

I. 问题的定义

项目概述

本项目将使用 Kaggle 提供的数据集信息，对 Rossmann 不同类型的药妆店进行分类预测。

Rossmann 是欧洲的一家药品连锁店，在 7 个欧洲国家拥有 3,000 家药店。Kaggle 比赛 Rossmann Store Sales 的本来要求是预测 Rossmann 未来的销售额，这里希望利用 Rossmann 药妆店的信息（比如促销，竞争对手，节假日）以及在过去的销售情况，训练模型以实现对 Rossmann 不同类型药妆店的分类。

这里采用 DecisionTree 模型对结果进行预测，并使用网格搜索（GridSearchCV）优化模型参数。

训练模型和进行预测所使用的数据集，是在 Kaggle 上下载的“train.csv”和“store.csv”，这里面分别包括了与 Rossmann 的销售额以及店铺信息有关的数据。

问题陈述

本项目要解决的问题是：根据数据集里提供的特征信息，对 Rossmann 店的店铺类型进行分类，属于监督学习范畴。因此，这里将数据集中“StoreType”一列作为标签，其余列作为用于判断标签的特征存在。

为实现这里的分类问题，需要对数据集进行以下处理：

- 首先需要对数据进行预处理，填补或丢弃缺失值和异常值；
- 对已经清理的数据集进行探索——虽然“StoreType”列出现在“stores.csv”文件中，但因为店铺类型也可能与销售信息相关，所以需要对这两个数据集都进行探索，并且需要进行数据集合并，以增加特征种类数量；
- 对合并的数据集提取标签集和特征集，对特征集里的非数字特征进行独热编码，之后将特征和标签集按时间顺序划分为训练集和测试集，用于训练模型。

使用训练好的模型对测试集进行预测，根据得分情况评估模型表现，最后优化模型参数，检验模型的鲁棒性。

评价指标

要衡量预测的效果，我们需要一个根据实际商店类型结果对预测进行打分的指标。因为我们对能够准确预测店铺类型感兴趣，因此我们使用准确率（accuracy_score）作为评价模型的标准是合适的。这里准确率计算的是对测试数据集（y_test）中预测正确的店铺数量占测试集样本总数的比例，即：

$$\text{accuracy\_score} = \frac{\text{y\_test 中预测正确的店铺数量}}{\text{y\_test 中的样本总数}}$$

II. 分析

数据的探索

本项目使用了 2 个数据文件，一个是包含销售信息的“train.csv”，另一个是包含商店信息的“store.csv”，这两个数据集中包含的具体字段信息如下：

表 1 数据集中包含的字段信息释义

字段名	释义
Store	每个商店独有的 Id
Sales	销售额
Customers	客流量
Open	商店是否处于营业状态
StateHoliday	国家假日类型. a 代表公休假日，b 代表复活节，c 代表圣诞节，0 代表不处于假日状态
SchoolHoliday	商店是否受学校放假的影响
Storetype	商店的类型，有 a,b,c,d 四种
Assortment	商店销售品的分类等级
CompetitionDistance	和最近竞争者的距离
CompetitionOpenSince[Month/Year]	最近的竞争者开张时大概的年/月
Promo	商店当天是否处于促销状态
Promo2	商店是否处于一个持续的促销状态中
Promo2Since[Year/Week]	店铺开始参与 Promo2 的年/月份
PromoInterval	Promo2 持续运行的间隔

首先查看对“train.csv”数据集中“Sales”一列的统计性描述：

表 2 数据集“Sales”列的统计性描述

count	1.017209e+06
mean	5.773819e+03
std	3.849926e+03
min	0.000000e+00
25%	3.727000e+03

50%	5.744000e+03
75%	7.856000e+03
Max	4.155100e+04

可以看到，“train.csv”数据集中共有 1017209 条数据。可以看到，销售额的最小值为 0，表明销售数据中存在已经关闭（“Open”== 0）的商店，又或者是数据中存在虽然店铺未关闭（“Open”== 1），但是销售额为 0 的异常值，因此需要对这些数据进行清理，以防止对预测产生干扰。

查看“store.csv”数据集，看是否在数据中存在空值：

表 3 “store.csv”数据集中的空值

Store	0
StoreType	0
Assortment	0
CompetitionDistance	3
CompetitionOpenSinceMonth	354
CompetitionOpenSinceYear	354
Promo2	0
Promo2SinceWeek	544
PromoInterval	544

可以看到，“Competition”系列和“Promo2”系列几个字段里有较多的缺失值，可以选择用中位数或者 O 填充空值，这里我们用 0 来进行填充。

对上面两个数据集均完成数据清理后，我们将其合并，并查看以商店类型分类依据的部分统计数据，其中“SMean”是人均消费额，“All”是每种商店的总数：

表 4 合并后的数据集信息统计

	Sales	Customers	SMean	All	PromoP	PromoP2
StoreType						
a	3165334859	363541431	8.706944	457042	44.75%	46.60%
b	159231395	31465616	5.060489	15560	38.16%	28.86%
c	783221426	92129705	8.501291	112968	44.91%	49.76%
d	1765392943	156904995	11.251350	258768	44.71%	56.95%

从表中可以得知，d 类型商店的人均消费是最高，尽管其总的销售额不如 a 商店多，而 a 类型则是是 Rossmann 数量最多的店铺类型。每种类型都有 50%左右的商店参加了两种促销活动，b 类型商店参加促销活动的商店占总数的比例最低。

探索性可视化

通过对数据的探索，了解到不同店铺类型的销售情况存在较大差异，所以可以查看不同店铺类型，其销售额随着月份变化的情况，同时考虑该店铺是否参加了促销的情况（“Promo”

== 0 or “Promo” == 1）以及是否处于持续促销的状态（(“Promo2” == 0 or “Promo2” == 1)）

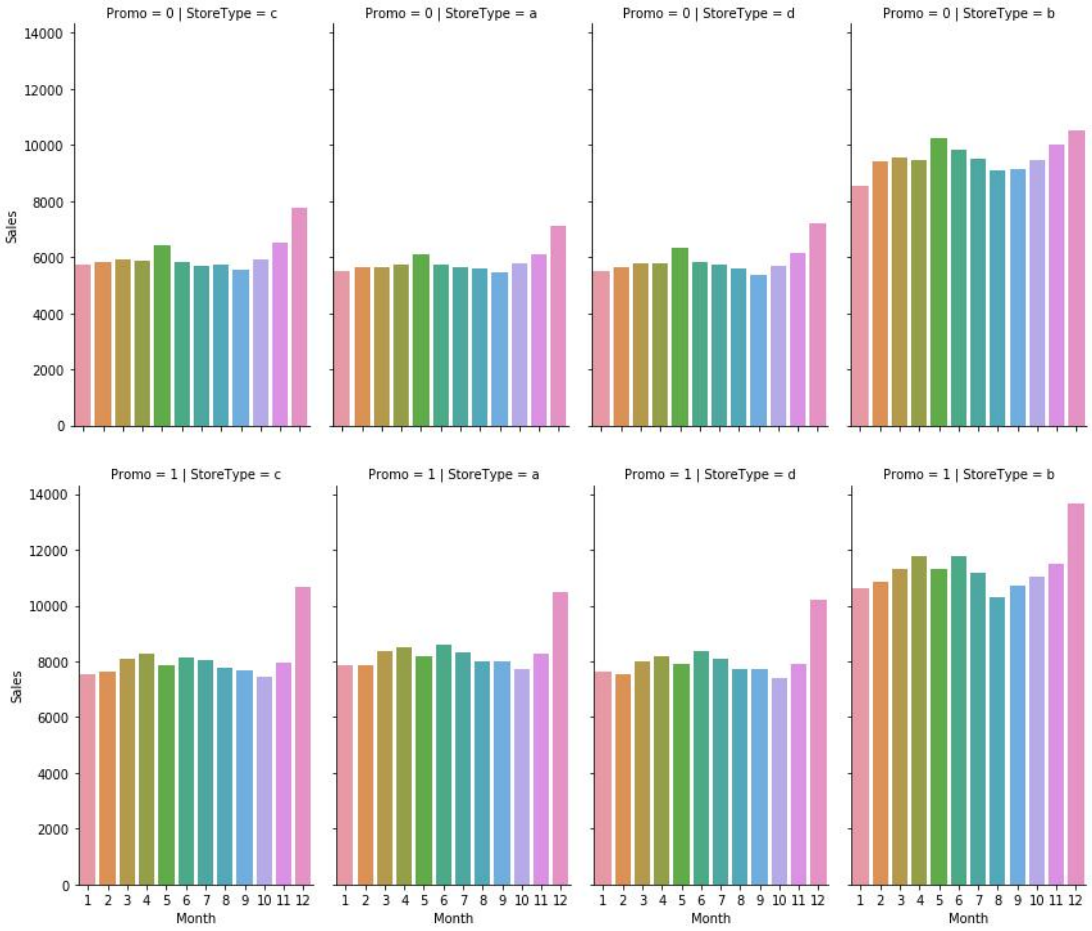
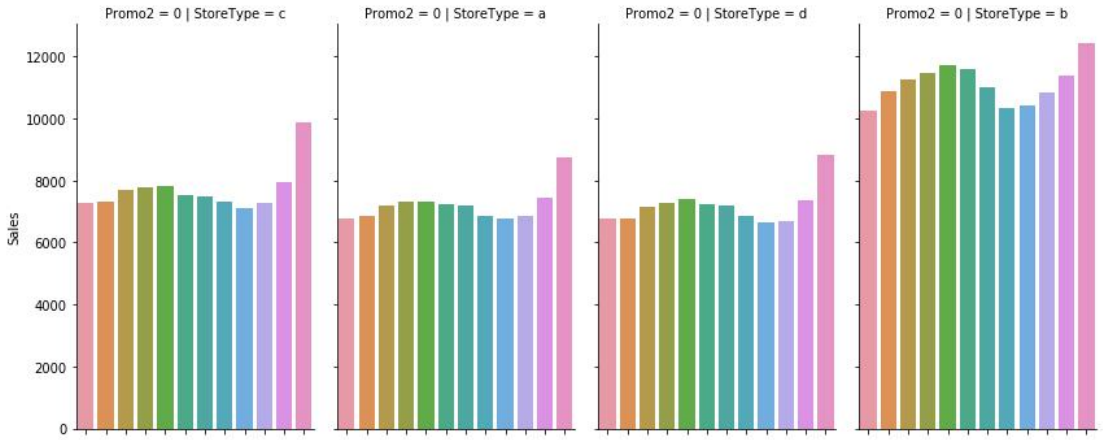


图 1 各类型商店参加/不参加促销每月的平均销售额情况

可以看到，所有店铺在有圣诞节的 12 月份的销售额都要比其余月份高，且 b 类型每个月的平均销售额明显比其他类型的商店高。参加促销的商店（“Promo” == 1）的每月的平均销售额要高于同类型的未参加促销的商店。



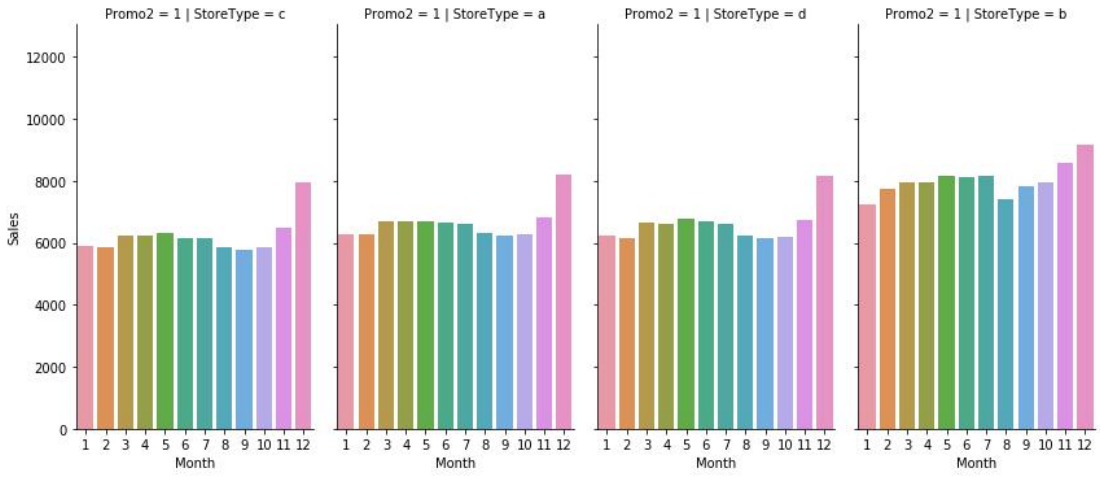
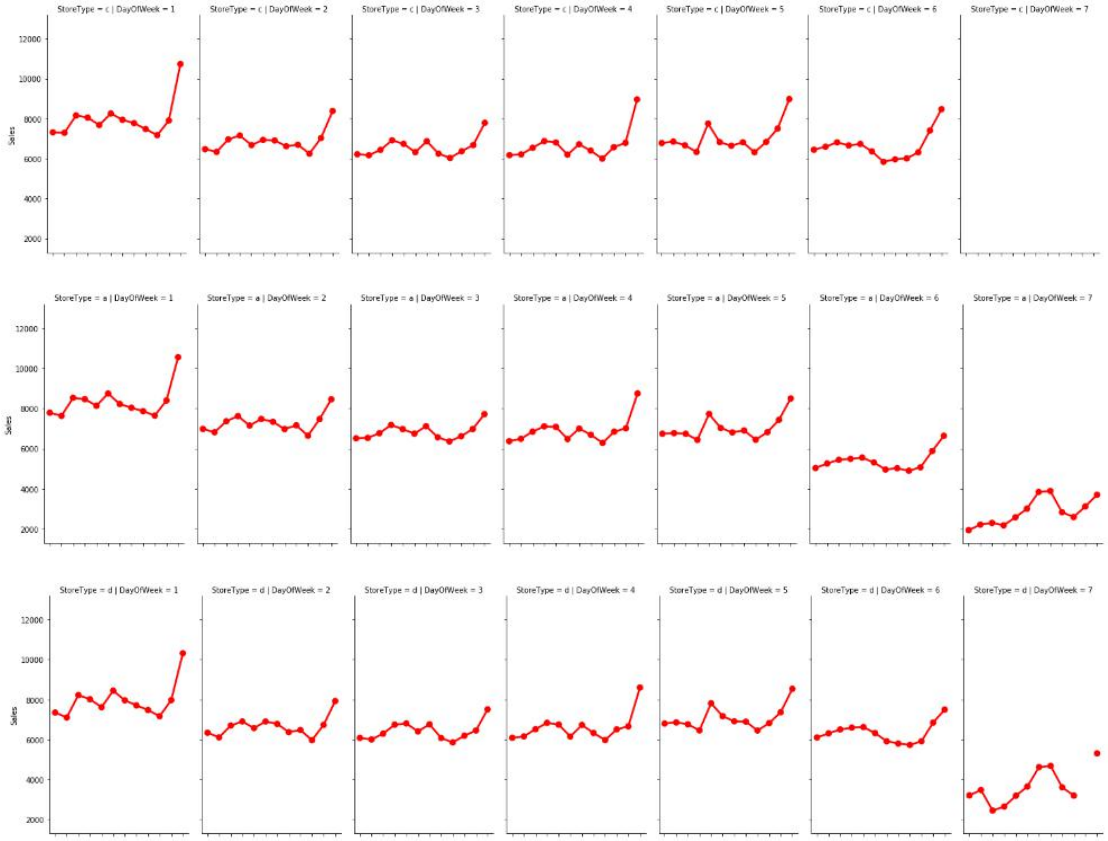


图 2 各类型商店参加/不参加持续促销每月的平均销售额情况

可以看到，不参加持续促销的店铺销售额明显高于同类型参与持续促销的店铺。这可能是因为参与持续促销的店铺周围有比较有竞争力的同行业店铺，同业竞争导致了店铺的销售额的下降。并且我们注意到，尽管 b 类型店铺每月的平均销售额最高，但人均销售额（见表 4）却最低，这说明该类型的商店可能主要经营高档产品，到该店铺的每位顾客未必一定会购买这里的商品。

再来看一下从周一到周日，各类型店铺的营业情况：



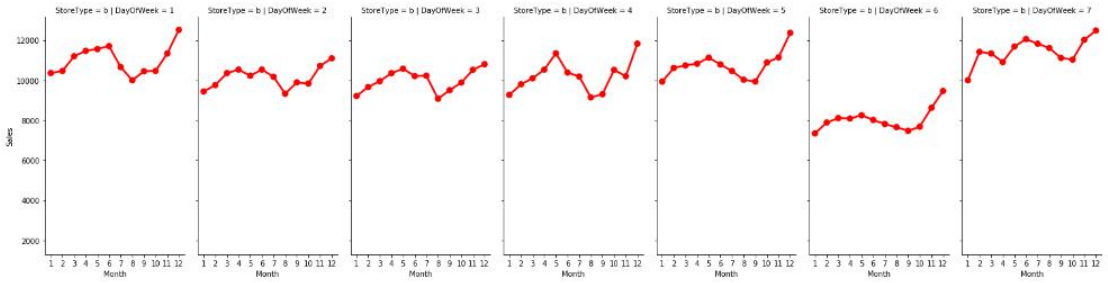


图 3 各个类型店铺周一到周日的营业情况统计

可以看到，很明显的，c 类型的店铺周末是不营业的，所以销售额为 0；a、d 周末的营业额降低较为明显，且 d 在每年的 11 月份的周末是不营业的；b 周末照常营业，且销售额相比其他几天更高。

算法和技术

首先，对于判断商店类型的分类问题，这里选用的模型是决策树 (DecisionTree)。决策树模型通过从数据特征中学习决策规则来训练模型，运行速度较快，并且，针对本项目中使用的训练数据，决策数模型还具有以下优势：

- 能够同时处理数据型和常规型属性；
- 在相对短的时间内能够对大型数据源做出可行且效果良好的结果；
- 对数据集中的缺失值不敏感；
- 可以处理不相关特征数据；
- 效率高，决策树只需要一次构建，反复使用，每一次预测的最大计算次数不超过决策树的深度；

在训练数据的准备过程中，需要处理缺失值和异常值，并提取标签和特征分别形成标签集和特征集，对提取的特征集进行独热编码。

之后，使用 sklearn 中的 train_test_split 将特征集和标签集随机地划分为训练集和测试集，用于训练 DecisionTree 模型，并用网格搜索算法优化模型参数。

基准模型

鉴于该分类问题是多分类问题（共有 4 种店铺类型），即对每一条要预测的数据，其属于任何一种店铺类型的概率为 1/4，即本项目采用 0.25 作为基准值，此为假设值，模型准确率得分越高则模型表现约好。

III. 方法

数据预处理

将 train.csv 原始数据中已经关闭的店铺，以及虽然未关店，但销售额为 0 的异常值进行了处理，将去除了这些值之后的数据存储为新的数据集 new_train_data 中：

```
new_train_data = train_data[(train_data['Open'] != 0) & (train_data['Sales'] != 0)]
```

对于 store.csv 原始数据中的“NaN”值，进行了填 0 的处理，并将处理后的数据放在新的数据集 new_store_data 中：

```
new_store_data = store_data.fillna(0.0)
```

之后，选取两个数据的交集进行合并：

```
train_and_store = pd.merge(new_train_data, new_store_data, how = 'inner', on = 'Store')
```

最后，对该数据集的“Data”列进一步处理，从中分解出了“Year”和“Month”这两列新的时间特征:

```
train_and_store['Year'] = train_and_store['Date'].apply(lambda x: int(x[:4]))
train_and_store['Month'] = train_and_store['Date'].apply(lambda x: int(x[5:7]))
```

执行过程

为节省训练时间，本项目首先从 train_and_store 数据集中随机抽样，得到样本数据，之后，将样本数据中的“StoreType”列作为标签集，将去掉“StoreType”列的其余数据作为特征集：

```
sample_data = train_and_store.sample(n=10000)
forecast = sample_data['StoreType']
features_raw = sample_data.drop(['StoreType', 'StoreType2'], axis=1)
```

之后对 features_raw 数据集进行独热编码：

```
features = pd.get_dummies(features_raw)
```

利用 train_test_split 数据拆分为训练集和测试集：

```
X_train, X_test, y_train, y_test = train_test_split(features, forecast, test_size=0.2, random_state=42)
```

首先使用默认参数建立模型，并使用训练集对模型进行训练。模型参数如下：

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

使用准确率得分 accuracy_score 用于评价模型性能，使用默认参数得到的模型得分为：

```
The test accuracy is 0.8705
```

完善

对决策树模型性能影响较大的超参数主要是_samples_split，max_depth 和 min_samples_leaf，

这里使用 GridSearchCV 对 min_samples_split 参数进行优化，优化前后模型得分分别为 0.8705，0.8710：

```
best_clf
------
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=4, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')

Unoptimized model
------
Accuracy score on test data: 0.8705

Optimized Model
------
Final accuracy score on the test data: 0.8710
```

IV. 结果

模型的评价与验证

通过对模型的参数进行网格搜索优化，得到了最终的模型，该模型对测试集预测的最终得分为 0.871，性能较好，与期待的结果一致。

进一步的，为验证模型的鲁棒性，我们用一条新的客户数据，用训练好的模型循环验证多次，看是否得到相同的预测结果：

```
Forcast Results
------
The No.1 time forecast result for the changed sample: ['d']
The No.2 time forecast result for the changed sample: ['d']
The No.3 time forecast result for the changed sample: ['d']
The No.4 time forecast result for the changed sample: ['d']
The No.5 time forecast result for the changed sample: ['d']
The No.6 time forecast result for the changed sample: ['d']
The No.7 time forecast result for the changed sample: ['d']
The No.8 time forecast result for the changed sample: ['d']
The No.9 time forecast result for the changed sample: ['d']
The No.10 time forecast result for the changed sample: ['d']
```

可以看到，连续预测 10 次的结果相同，表明模型对于该问题足够稳健可靠，也说明模型得出的预测结果是可信的。

合理性分析

该模型对测试集预测的最终得分为 0.871，远高于基准得分 0.25，表明该模型较好的解决了对商店类型预测的问题。

V. 项目结论

结果可视化

这里绘制出的热图给出了商店类型与各个特征之间相关性的强弱，其中“StoreType2”是表示商店类型的字段：

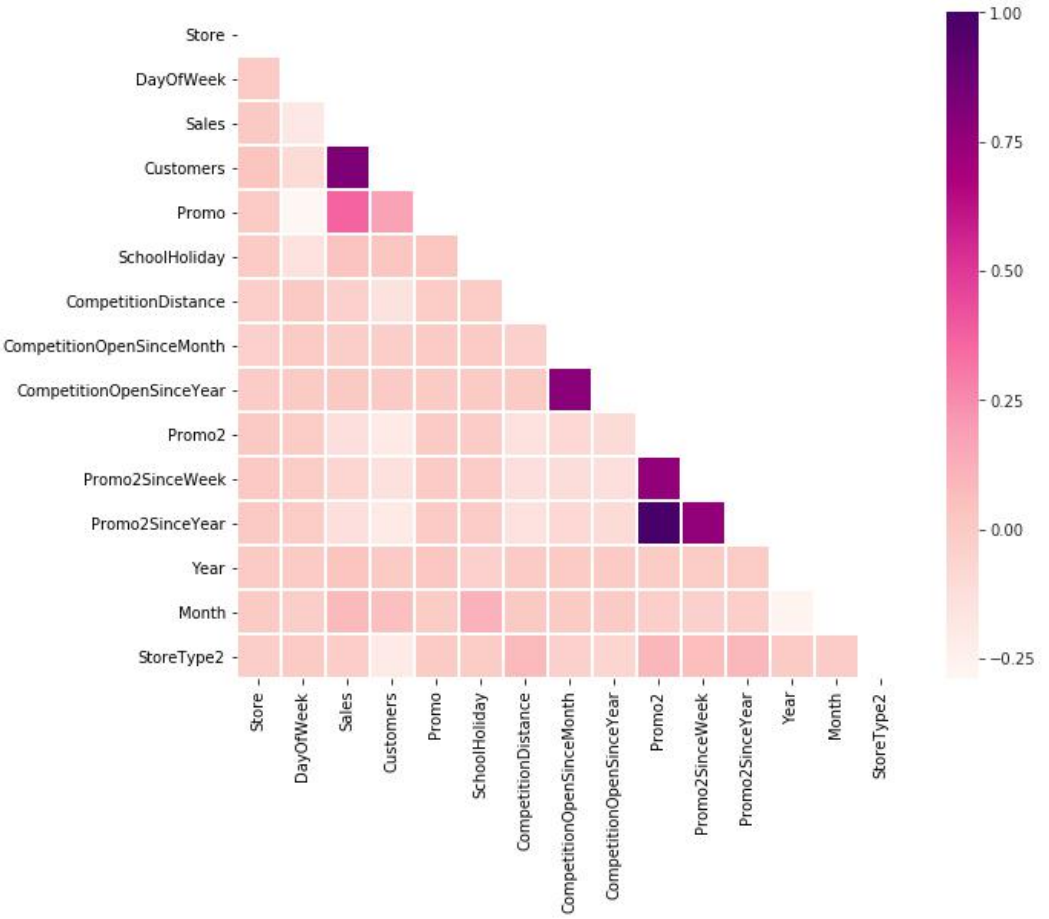


图 4 各特征间相关性热图

从图中可以看到，除了销售额，参与促销的情况，以及每周不同的时间外，商店类型还可能与其他特征如“CompetitionDistance”，“Promo2”以及“Promo2SinceYear”等具有较强的相关性。所以，学习器可以从这些特征中学习并对商店类型给出预测。

对项目的思考

本项目比较顺利的解决了了对商店类型判断的问题，预测的准确率也较高。这个项目的最有意思的地方是自己尝试了整个数据分析的流程，从提出问题，到数据的收集和清理，在到数据探索，最后根据数据建模和预测。

主要困难主要出现在数据探索阶段，由于对 Python 语言熟悉度不够，在对数据进行格式处

理（如将“Date”分解成新的特征）和进行数据可视化时花费了比较多的时间完成，选取的可视化的方式可能也不是最好的表现方式。在利用网格搜索进行模型优化时，如果参数选取不善反而遇到了模型性能裂化的问题，对于模型参数的熟悉程度不够。

需要作出的改进

在数据处理阶段，对于 NaN 值的处理是直接填充 0，没有尝试填充中位数的方案，可能采用后一种方案会得到更好的预测效果；

模型的选取上只是使用了决策树模型，这种模型在训练过程中容易产生过拟合的问题，可以尝试使用 AdaBoost 或者 SVM 等模型，并比较不同模型的训练结果。

参考资料

Kaggle 竞赛项目——Rossmann 销售预测 Top1%
20 Newsgroup Document Classification Report
Stackoverflow
Time Series Analysis and Forecasts with Prophet
维基百科