“基于历史销售数据预测销售情况”项目报告

1. 数据获取及预处理

1.1 数据来源

本项目将使用kaggle中的competitive-data-science-predict-future-sales数据集，数据集中包含2013年1月至2015年10月的每日历史销售数据和这些商店和产品在2015年11月的销售额，分别可以作为训练集和测试集。

1.2 数据说明

sales_train.csv中是2013年1月至2015年10月的每日历史数据，可以作为训练集；test.csv为2015年11月的销售额，可作为测试集。

各个字段及其含义为：

- ID- 代表测试集中的（商店，商品）元组的ID
- shop_id-商店的唯一标识符
- item_id-产品的唯一标识符
- item_category_id-产品类别的唯一标识符
- item_cnt_day-销售的产品数量。预测该指标的每月金额
- item_price-商品的当前价格
- 日期 -格式为dd / mm / yyyy的日期
- date_block_num-连续的月份号。2013年1月为0,2013年2月为1，...，2015年10月为33
- item_name- 项目名称
- shop_name-商店名称
- item_category_name-项目类别名称

1.3 数据预处理

首先读取数据

In [ ]:

```
sales_train = pd.read_csv('competitive-data-science-predict-future-sales/sales_train.csv')
test = pd.read_csv('competitive-data-science-predict-future-sales/test.csv')
submission = pd.read_csv('competitive-data-science-predict-future-sales/sample_submission.csv')
items = pd.read_csv('competitive-data-science-predict-future-sales/items.csv')
item_cats = pd.read_csv('competitive-data-science-predict-future-sales/item_categories.csv')
shops = pd.read_csv('competitive-data-science-predict-future-sales/shops.csv')
```

检查数据集中的缺失值，发现无缺失值。

```
date              0
date_block_num    0
shop_id           0
item_id           0
item_price        0
item_cnt_day      0
dtype: int64
```

检查测试集中的所有商店和物品是否也在训练集中

In [ ]:

```
test_shops = test.shop_id.unique()
train = sales_train[sales_train.shop_id.isin(test_shops)]
test_items = test.item_id.unique()
train = train[train.item_id.isin(test_items)]
```

去除训练集中重复的部分

In [ ]:

```
print('Before drop train shape:', sales_train.shape)
sales_train.drop_duplicates(subset=['date', 'date_block_num', 'shop_id', 'item_id', 'item_cnt_day'],
                    keep='first', inplace=True)
sales_train.reset_index(drop=True, inplace=True)
print('After drop train shape:', sales_train.shape)
```

```
Before drop train shape: (2935849, 6)
After drop train shape: (2935825, 6)
```

转换为日期格式

In [ ]:

```
parser = lambda date: pd.to_datetime(date, format='%d.%m.%Y')
```

为了加快后续"模型"的训练速度，对所有数据进行标准化处理

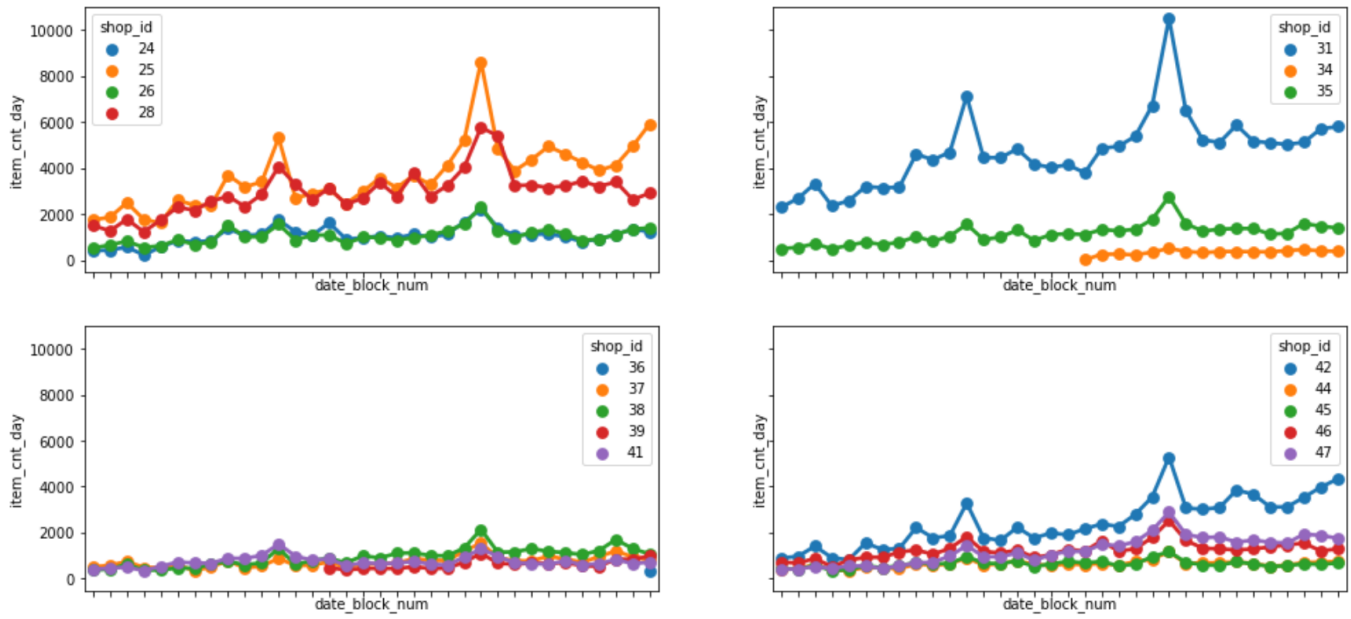
In [ ]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
to_drop_cols = ['date_block_num']
feature_columns = list(set(pre_cols + index_cols + list(all_set)).difference(to_drop_cols))
all_set[feature_columns] = sc.fit_transform(all_set[feature_columns])
```

2. 数据分析与可视化

2.1 随时间增长销量的变化

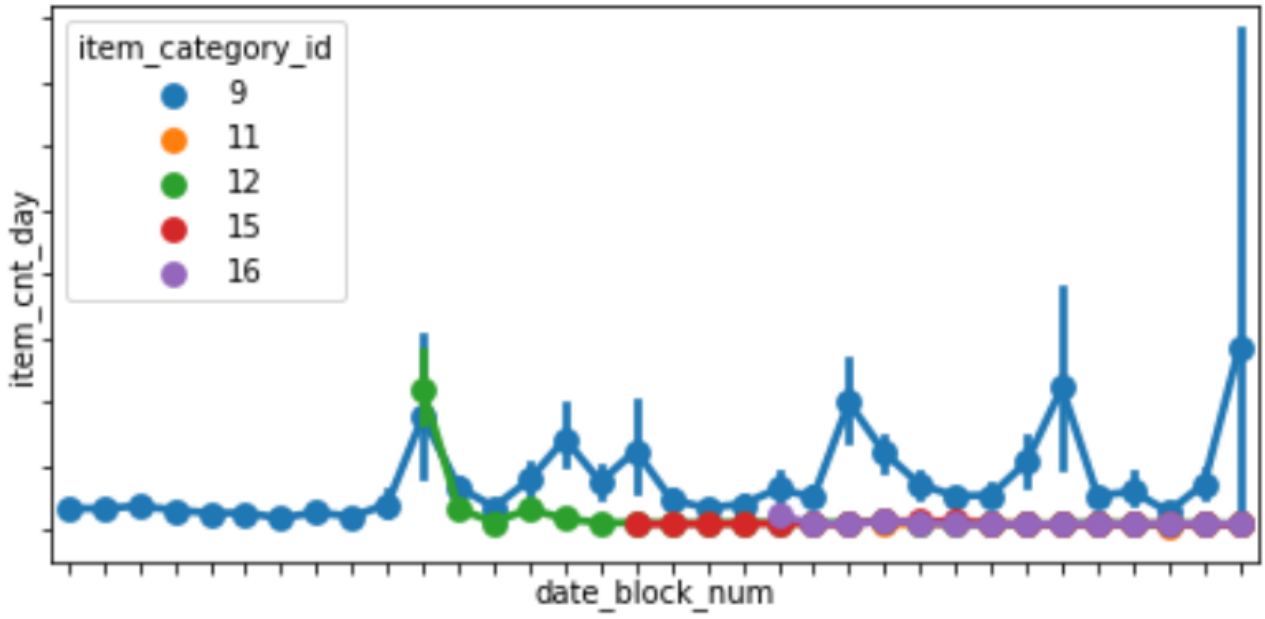
首先，观察各个商店随时间增长商品销量的变化



从图中，我们可以发现：

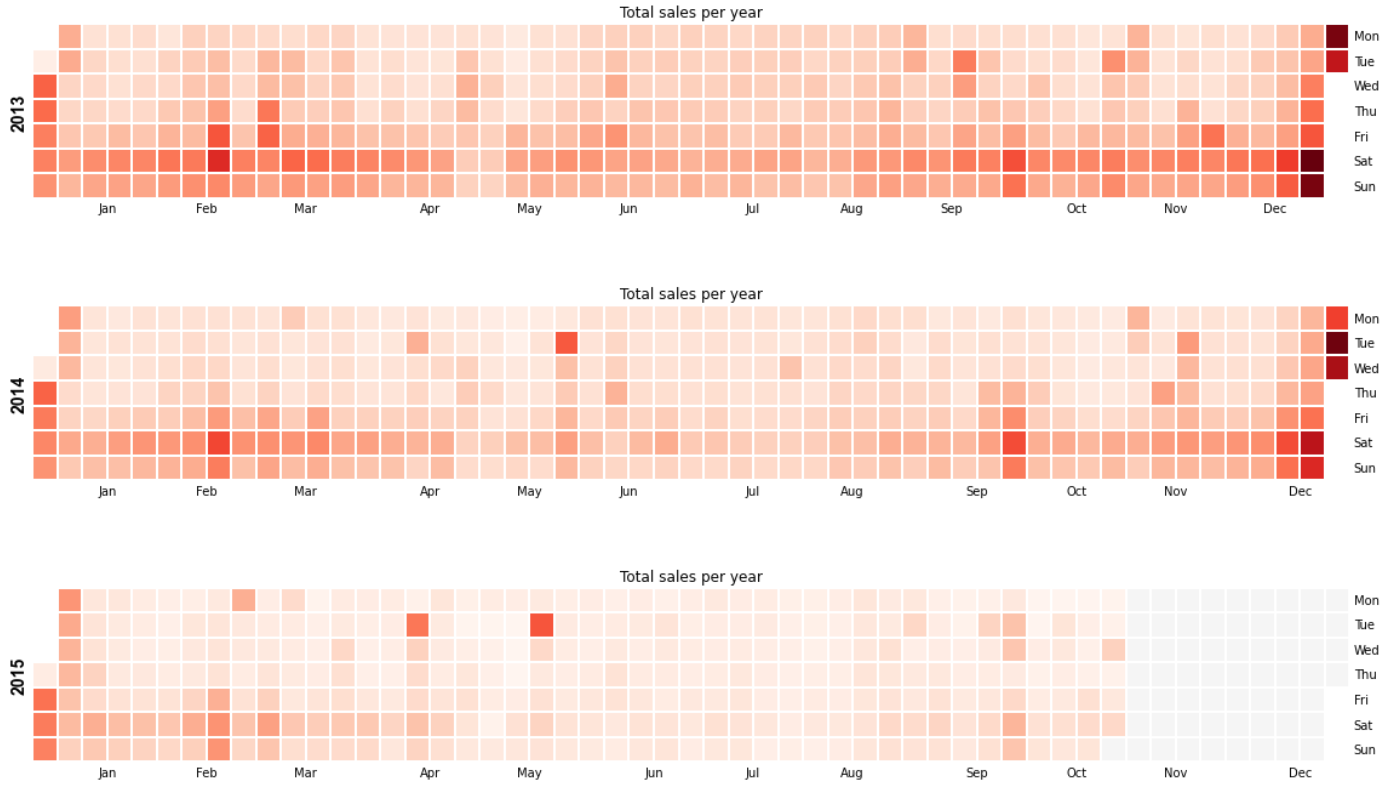
总体来说，在每年年初的时候，各个商店的销量会有明显的升高。这其中很可能有人们休假的影响。随着假期的到来，商品销量明显变高。

除此之外，还可以来观察随时间增长各类商品销量的变化



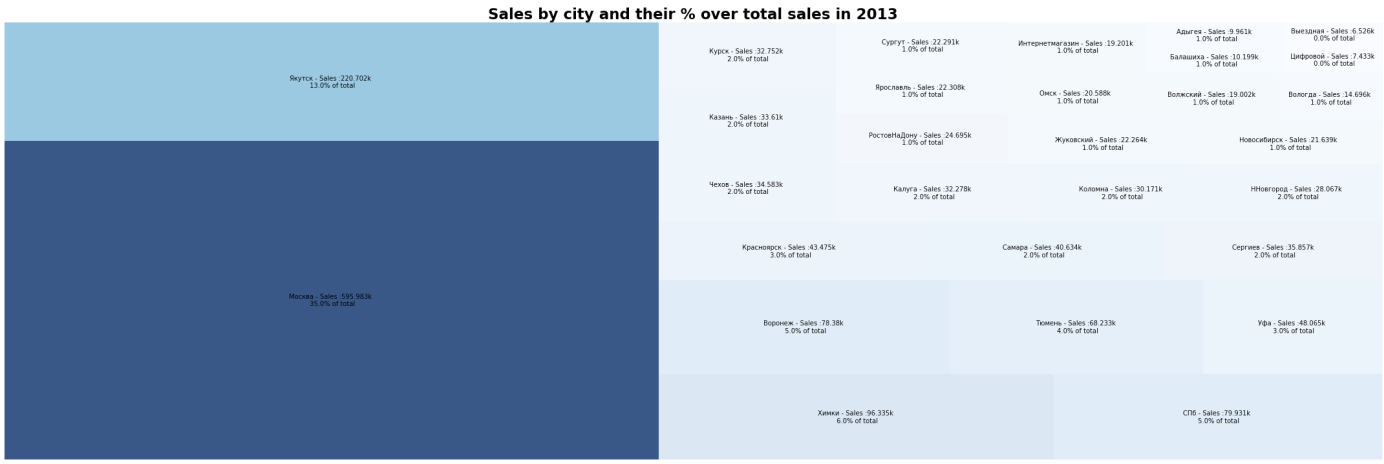
可以看到，从2014年起，id为9的商品类销量明显上升。

2.2 每年总销售热力图



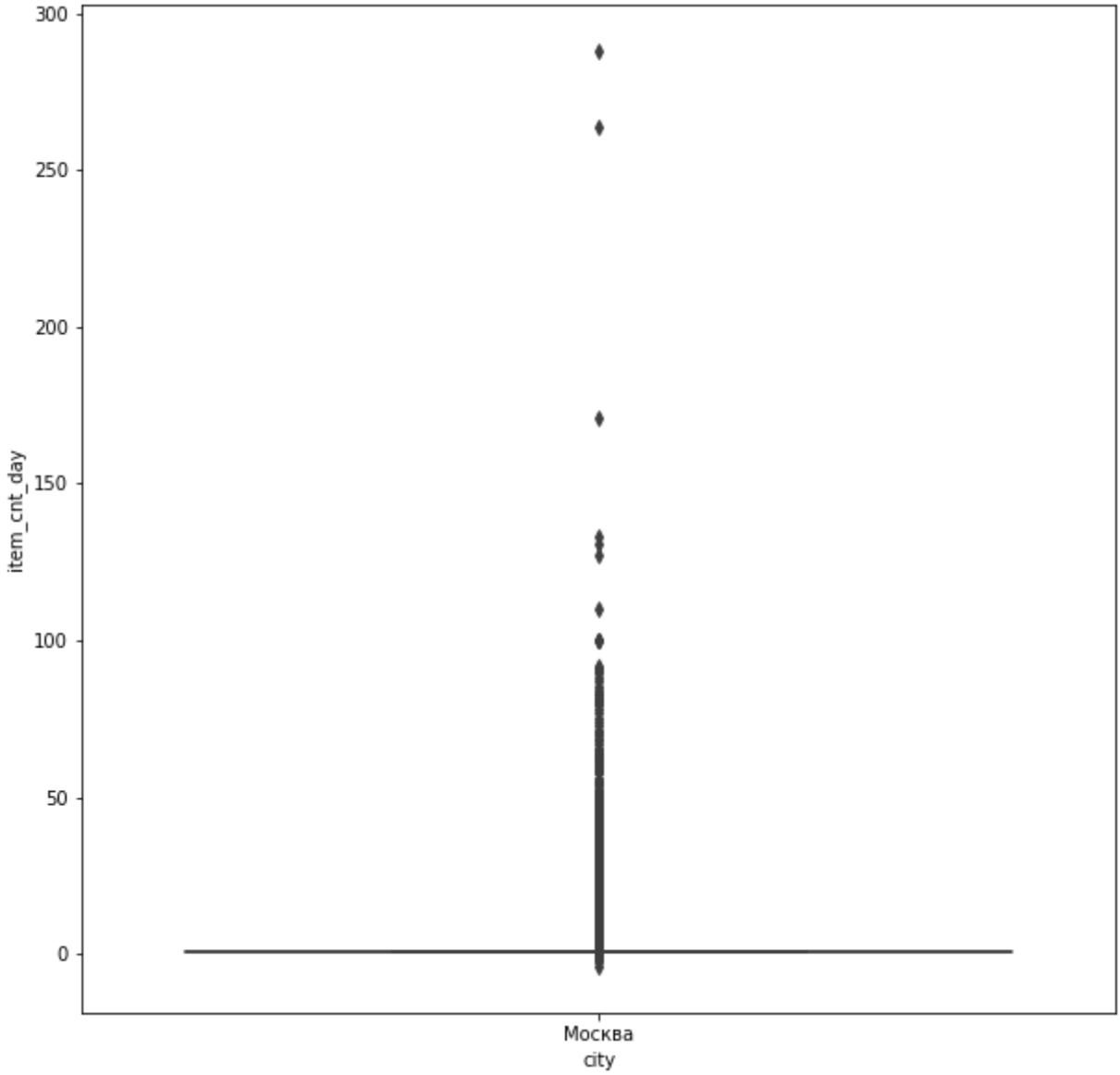
通过销售热力图可以看到一年之间各个月份各每周各天的销售情况，可以得到在周末的销售数量一般处于一周之中的最高部分。一年中，年中之间的销售是一年中最淡的时候。

2.3 各城市销售情况



以上分别为2013年按城市划分的销售额及其占总销售额的百分比。可以看出俄罗斯首都莫斯科以绝对的大比例3年都是第一；雅库茨克位列第二；其余各城市均占比在5%以下。

使用盒图查看莫斯科市的销售数据：



可以看到莫斯科市销售数据的巨大差异。

3. 模型选取

3.1 选取原因

GBDT模型：是一种基于集成思想的决策树模型，GBDT采用加法模型，通过不断减小训练过程产生的残差，以此对数据进行回归或分类。在大数据集上的表现良好，运行速度快，准确率较高，实现简单且鲁棒性好。

SGDRegressor模型：是使用随机梯度下降进行线性回归的，随机梯度下降是不将所有样本都进行计算后再更新参数，而是选取一个样本，计算后就更新参数。随机梯度下降法每次用一个样本调整参数，逐渐逼近，效率高。

LinearRegression模型，利用线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析，计算简单，结果易于理解。

3.2 训练模型

训练GBDT模型：

In [ ]:

```
lgb_params = {
              'feature_fraction': 0.75,          # 每次迭代的时候随机选择特征的比例，默认为1，训练前这
              'metric': 'rmse',                  # root square loss(平方根损失)
              'nthread':1,                      # LightGBM 的线程数
              'min_data_in_leaf': 2**7,          # 一个叶子上数据的最小数量. 可以用来处理过拟合
              'bagging_fraction': 0.75,          # 类似于 feature_fraction, 但是它在训练时选特征
              'learning_rate': 0.03,            # 学习率
              'objective': 'mse',               # regression_l2, L2 loss, alias=regression, mean_squar
              'bagging_seed': 2**7,              # bagging 随机数种子
              'num_leaves': 2**7,               # 一棵树上的叶子数
              'bagging_freq':1,                  # bagging 的频率, 0 意味着禁用 bagging. k意味着每 k次
              'verbose':1                       # verbose: 详细信息模式，0 或者 1
              }
```

训练SGDRegressor模型:

In [ ]:

```
sgdr= SGDRegressor(
    penalty = 'l2' ,            # 惩罚因子：L2正则化
    random_state = 0
    )
```

训练LinearRegression模型:

In [ ]:

```
lr = LinearRegression()
estimator = lr
estimator.fit(train_set, train_value)
```

4. 实验结果

4.1 实验结果

利用训练出的模型来预测2015年11月的销售数据。

Out[43]:

	ID	item_cnt_month
0	0	1.106
1	1	1.006
2	2	1.009
3	3	1.011
4	4	1.005
5	5	1.007
6	6	1.939
7	7	1.035
8	8	2.048
9	9	0.000
10	10	3.311
11	11	0.997
12	12	1.008
13	13	1.011
14	14	2.115
15	15	3.084
16	16	0.000
17	17	1.004
18	18	2.058

4.2 评估

采用均方根误差（RMSE），来进行测试样本中的误差评估，可以发现在测试集上，SGDRegressor模型和

LinearRegression模型的效果比GBDT模型表现较好一些。

```
Train RMSE for lightgbm is 0.442378
Test RMSE for lightgbm is 0.138512
Train RMSE for SGDRegressor模型 is 0.583812
Test RMSE for SGDRegressor模型 is 0.000000
Train RMSE for LinearRegression模型 is 0.554747
Test RMSE for LinearRegression模型 is 0.000084
```

5. 存在的问题

数据预处理的方式较为简单，对于异常值简单的丢弃，可能造成数据缺失和模型鲁棒性降低等情况，可以尝试一些其他的策略，比如：异常值替换为认为合理的区间的端点值；连续变量根据分位数或人工限定阈值离散化变为类别变量，消除极端值的影响等等方式。

6. 任务分配

- 白璐： 数据处理、数据分析、文档编写、训练模型
- 胡玉龙：训练模型、文档编写、数据处理
- 易晗：文档编写
- 田君玉：数据分析、文档编写
- 范晟杰：数据分析可视化、文档编写