Rossmann Sales Prediction with Supervised Learning

Mengfei Li(李梦非)

Oct 02, 2017

1 Definition

1.1 Project Overview

机器学习在实现商业预测的领域中，已经越来越被广泛地使用，并且已经有了很多成功的案例[1]。这个毕业设计针对要解决的问题是 Rossmann 商店的销售预测。针对这个毕业项目,Kaggle 上给出的数据[2]都是基于真实统计的结果。首先简要的介绍下关于 Rossmann 的相关背景知识。Rossmann 是德国一家已销售洗化用品的超市，也提供一些非处方药物，和其他的一些日用品，母婴产品等[3]。Rossmann 是一家相对比价大规模的此类型超市，在德国有 1115 家连锁店，在整个欧洲范围以内有超过 3000 家连锁, 其主要的竞争对手是德国的另一家连锁超市 dm[4]。

那么什么因素会影响商店的销售额呢? 对于一个非经济专业毕业的人士来看，首先能被想到的几个因素会是：商店类型，地理位置，销售物品，当天逛商店的总人数以及商店是否在搞促销活动等等。尽管网络上有一些经济学的资料可以查阅，但是对于缺少领域知识的其他专业人士来说，并没有一个非常详细和直观的概念。因此，机器学习在这里就能派上用场。Kaggle 上给出的数据来看，可以被用来作为特征 (feature) 来预测标签 (label)[5]包括一些容易被理解的数据，比如：商店类型 (StoreType), 品种分类 (Assortment)，促销信息 (Promo) 等等。同时，数据中也包含了一些不那么容易直接被理解为可以和销售额挂钩的数据，至少从概念上没有直接的联系，比如：竞争开始的月份 (CompetitionOpenSinceMonth)，促销的月份 (PromoInterval) 等等。

机器学习可以主要分为监督学习 (Supervised Learning), 非监督学习 (Unsupervised Learning) 和半监督学习 (Semi-supervised Learning) 这几大类 (Brownlee, 2016)。此外，增强学习 (Reinforcement Learning) 和深度学习也是机器学习所涵盖的范畴 (Sutton and Barto, 1998; LeCun et al., 2015)。这个毕业项目–Rossmann Sales Prediction 中，需要预测的标签已经给出，为: 各商店在未来某一段时间的销售额。应此这个项目需要解决的问题属于机器学习当中监督学习的范围。更加具体的来说，属于监督范围内的回归 (Regression) 问题。这个项目的核心是通过机器学习建立相关的预测模型，最终需要解决通过将特征数据输入这个模型，从而得到商店销售额的预测值作为输出。

[1] https://zhuanlan.zhihu.com/p/25725004

[2] https://www.kaggle.com/c/rossmann-store-sales

[3] https://www.rossmann.de/einkaufsportal.html

[4] https://www.dm.de/

[5] 在这里, 标签指的是商店的销售额；特征指的是用来预测销售额的数据。

1.2 Problem Statement

在这个项目里，监督学习会被用来建立预测模型实现对 Rossmann 的销售额进行预测。通过所给出的 1115 家 Rossmann 零售店的历史销售记录来预测出这些商店在未来某个时间段的销售额。在这个项目中，标签对应的是数据集中的"Sales"，也就是需要预测的数据。其他所有的特征数据则需要经过相应的处理才可以被机器学习算法所使用。原因主要是：并不是所有的数据都是完整的和可以直接被使用的，其中有缺失的数据和需要合并的数据。除此之外，在开题报告中已经讨论，这个项目还需要运用一定程度的特征工程 (Feature Enginnering)，其中包括特征的选择，提取，合并等。

在特征工程被完成之后则需要对数据集进行拆分处理，分割成训练集 (Training Set)，验证集 (Validation Set) 和测试集 (Testing Set)。之后使用 Xgboost[6]来建立机器学习模型 (在开题报告中已经给以阐述)。最终希望得到一个高精度和运算时间可以接受的模型。

关于验证所建立模型的预测能力是否符合预期，则考虑将预测出来的结果上传至 Kaggle 竞赛页面[7]，参考所处的排名位置来确定自己建立的模型是否满足预期。关于评价的基准，在开题报告中我给自己的预期是希望能够进入前 30% 的排名，这个排名可以表示所建立的模型基本符合预期。之后再通过对 Xgboost 算法进行参数优化以及适当的模型组合 (最简单的组合方法是对多个模型预测的结果求平均值[8])，来实现更高的排名，希望可以最终进入前 15% 的排名。

1.3 Metrics

对于预测结果的的评价标准，在 Kaggle 上所给出的是计算预测值和真实值之间的根均方误差百分比 (Root Mean Square Percentage Error) – RMSPE。计算方法由以下计算式 (1) 给出:

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{y_i - \hat{y}_i}{\hat{y}_i})^2} \tag{1}$$

在以上公式中，y_i 对应的是一个商店一天所对应的真实的销售值；$\hat{y}_i$ 所对应的是一个商店一天所对应的预测值[9]。随后的评分排名也是基于这个公式的计算结果的在这个项目中，最终的真实值是不被公布的。在正式上传之前，预测出来的结果需要按照"sample_submission" 所给出的格式进行整理。接下来系统会自动对其进行 RMSPE 的计算，并给出评分。

2 Analysis

2.1 Data Exploration

这个项目所对应的数据集可以在 Kaggle 官网下载，下载链接在脚注中给出[10]。给出的数据一共包括以下四组:"sample_submission","store.csv","test.csv" 和"train.csv" 四组数据。其中,"sample_submission" 的作用在上一节已经给以说明，作用是提供一个上传文件的标准。其余的三组则是对于机器学习来说非常重要的数据集。首先对这几组数据做一个简要的分析。

[6] https://github.com/dmlc/xgboost

[7] https://www.kaggle.com/c/rossmann-store-sales/leaderboard

[8] http://blog.csdn.net/google19890102/article/details/46507387

[9] https://www.kaggle.com/c/rossmann-store-sales/details/evaluation

[10] https://www.kaggle.com/c/rossmann-store-sales/data

”train.csv” 是用来训练机器学习模型的数据集。在这个项目中，需要预测的的数据 (标签) 是商店的销售额”Sales”, 其他的数据则 (特征)，则是被用来预测商店的销售额。”test.csv” 是一个独立的测试集，是用来最终评测模型得分的。预测出来的结果必须上传至 Kaggle 才能看到评分，因为”Sales” 的真实值是被隐藏的。至于”store.csv”，这是一个统计了 1115 家 Rossmann 零售店的其他的一些基础的相关特征，这个数据集与时间序列无关。这个数据集在之后的处理中需要和”train.csv” 以及”test.csv” 合并。

以下是对数据集中的所有特征做一个简要的说明：

- **Id:** 编号，独立于商店编号，包含了日期信息。
- **Store:** 商店编号。
- **Sales:** 每日销售额。这个特征要单独提取出来作为预测的 label。
- **Customers:** 每天改商店的顾客数量。
- **Open:** 当天是否营业。0 表示关闭，1 表示营业。
- **StateHoliday:** 当天是否为法定假日。a 表示公共假日,b 表示复活节,c 表示圣诞节,0 表示当天不是法定假日。
- **SchoolHoliday:** 当天是否是学校假期。
- **StoreType:** 商店不同的类型，类型分别用 a,b,c,d 来表示。
- **Assortment:** 商品类别，分别用 a,b,c 来表示。
- **CompetitionDistance:** 该商店最近的一个竞争者的距离，单位米。
- **CompetitionOpenSince[Month/Year]:** 该商店最近的一个竞争者从何时开业的。
- **Promo:** 当天是否在促销。
- **Promo2:** 商店是否参与持续促销。
- **Promo2Since[Year/Week]:** 在哪一年，哪一周，该商店参与持续促销。
- **PromoInterval:** 促销活动所在的月份。

在此之后，对数据做一些基础的可视化，在这里，只是大致分析一下数据集的统计情况，包括数据的缺失程度，异常值等等。在下一个 Section 中，将详细的对数据进行探索性分析 (EDA) 以及相应的可视化。图1显示所有商店被统计次数的直方图。

通过观测，可以发现，所有商店被统计的频率次数都是非常接近的，肉眼观察，每个商店的销售情况都被统计到 9000 次以上。如此看来，有足够的数据可以被用来进行机器学习。同时通过该直方图可以看出，每个商店被统计的次数并不是相等的，也就是说，这里面有可能存在不少的缺失数据，在之后的数据处理中需要注意。

从图2可以看出，当所有的商店按照不同的工作日进行统计的话，一周内 7 天相对应的统计次数都是非常接近的。

接下来，再对所有商店的销售额做一个可视化，如图3所示。

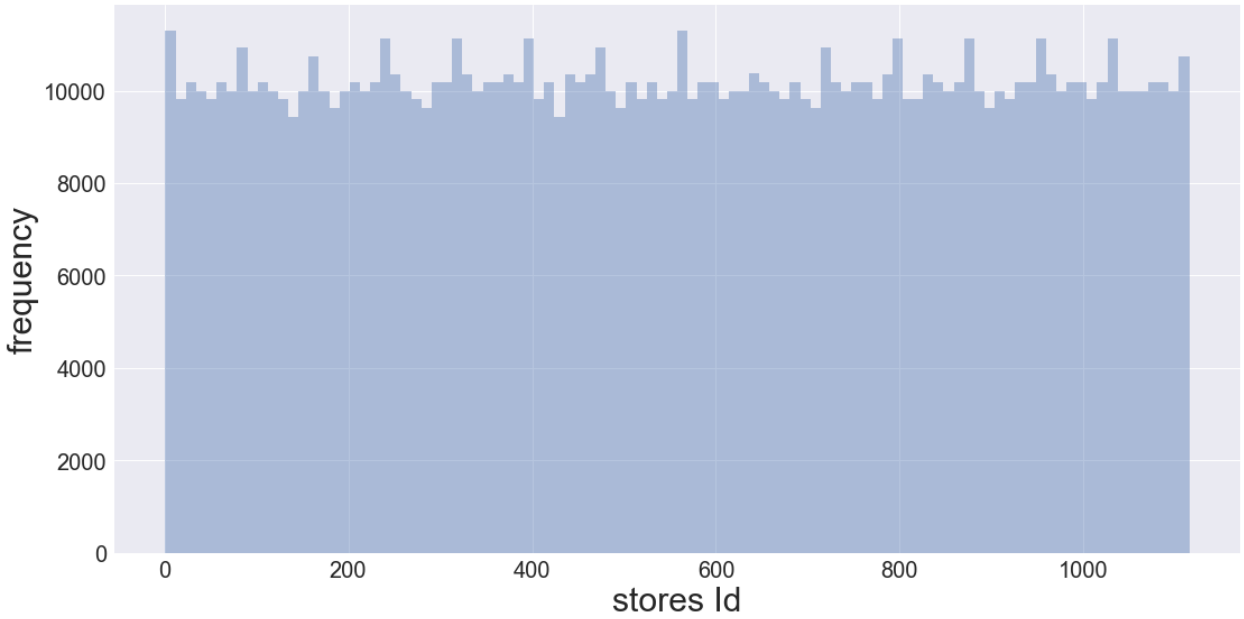

图 1: 所有商店被统计的次数-按照商店 ID 进行统计的直方图

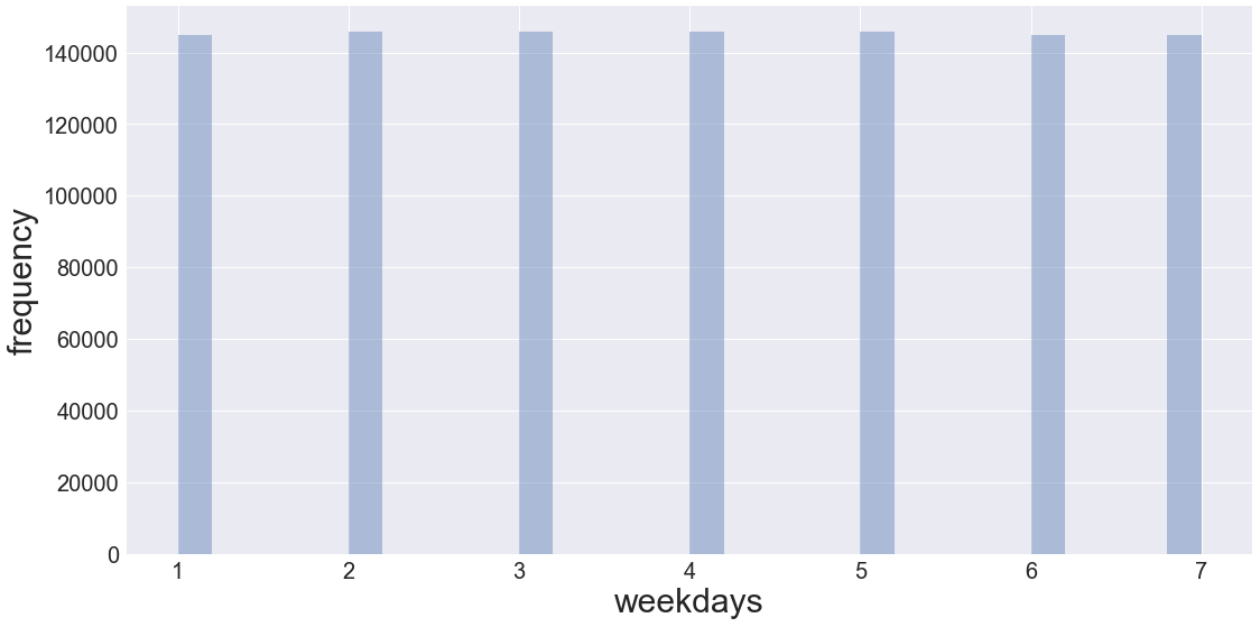

图 2: 所有商店被统计的次数-按照工作日进行统计的直方图

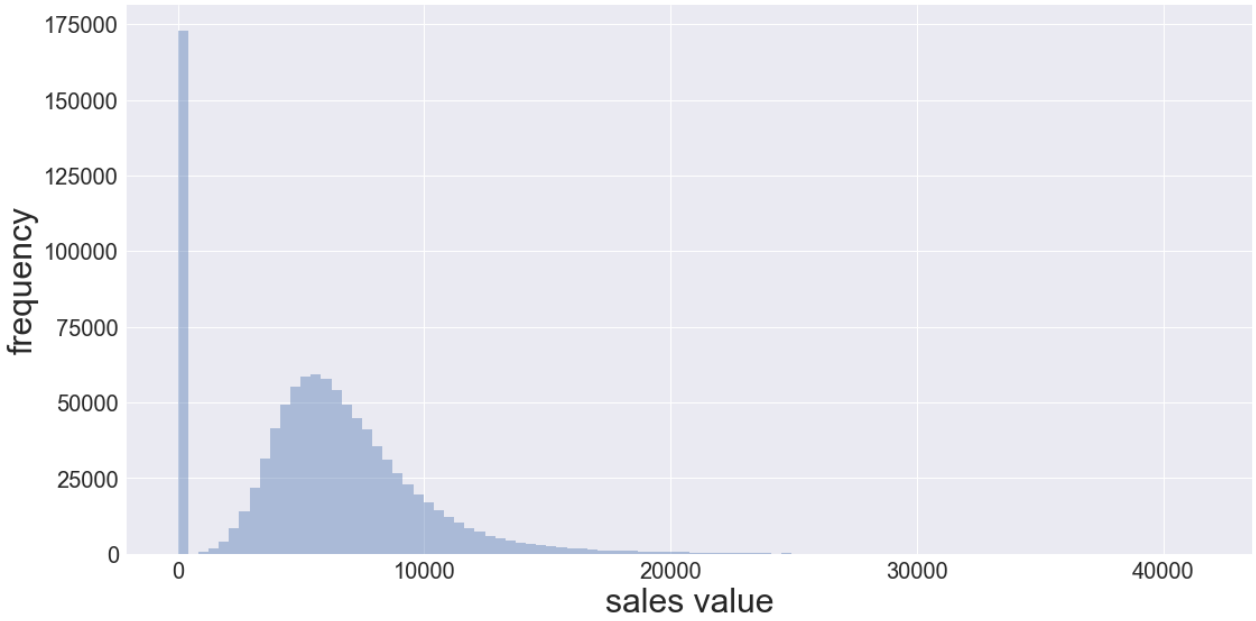

图 3: 所有商店销售额的直方图

通过图3可以观测出，出现最多的销售额为 0。那是因为周日和假日商店的销售额均为 0，出现的频率最高。其余剩下来的销售额数据的分布情况，则非常接近正态分布。使用 pandas 导入 *test.csv*, 发现这是一个 41088×8 的 DataFrame。相对于 *train.csv* 来说，多出一组数据是 Id(编号)，少了两组数据是 Customers 和 Sales。因为这两组数据都是需要过了当天才能统计的数据，所以没有出现在 *test.csv* 中。在这个数据集中，相对于其他统计的数据，Open 设个数据一共有 11 个缺失值，数量为 41077。

所有的商店开业与否相对当天 (工作日) 的销售图由图4所显示。从这个图可以简单看出，在统计不营业状态下, 所有商店的销售额均为 0，其中不存在异常值 (Outlier)；在统计商店工作日营业的状态下，大多数的销售额均处在中间平均值的附近，最大值和最小值和平均值相距很远，其中存在异常值。而这些异常值在开始使用机器学习建立预测模型之前，应当予以剔除。

至于"store.csv"，则存在着更多的缺失数据。CompetitionDistance 缺失了 3 个商店的数据；CompetitionOpenSinceMonth 和 CompetitionOpenSinceYear 缺失了 354 个商店的数据;Promo2SinceWeek,Promo2SinceYear 和 PromoInterval 缺失了 544 个商店的数据。在开始机器学习之前，要对这部分缺失的数据进行处理。

2.2 Exploratory Visualization

通过这些探索性的可视化，可以对数据有一个直观的认识。在之前的 Section 中，图1,3,4分别描述了所有商店被统计的次数，商品销售额分布情况以及销售额对应工作日/开业与否的情况。以上只是一些基本的可视化。这个章节中, 更加详细的数据集的探索性分析 (以可视化的形式) 将会被分析和讨论。

首先，我们可以分析一下促销与否与销售额之间的关系。在这里，仍然是使用 Violinplot 的格式，工作日也被考虑进去，如图5所示。

通过以上这幅图, 我们可以看出在商店搞促销活动的时候 (对应的颜色为绿色)，在五个工作日当中，商店的销售额明显大于商店在不搞促销活动的时候 (对应为蓝色)。由此可以分析出，Promo 应该是影响预测结

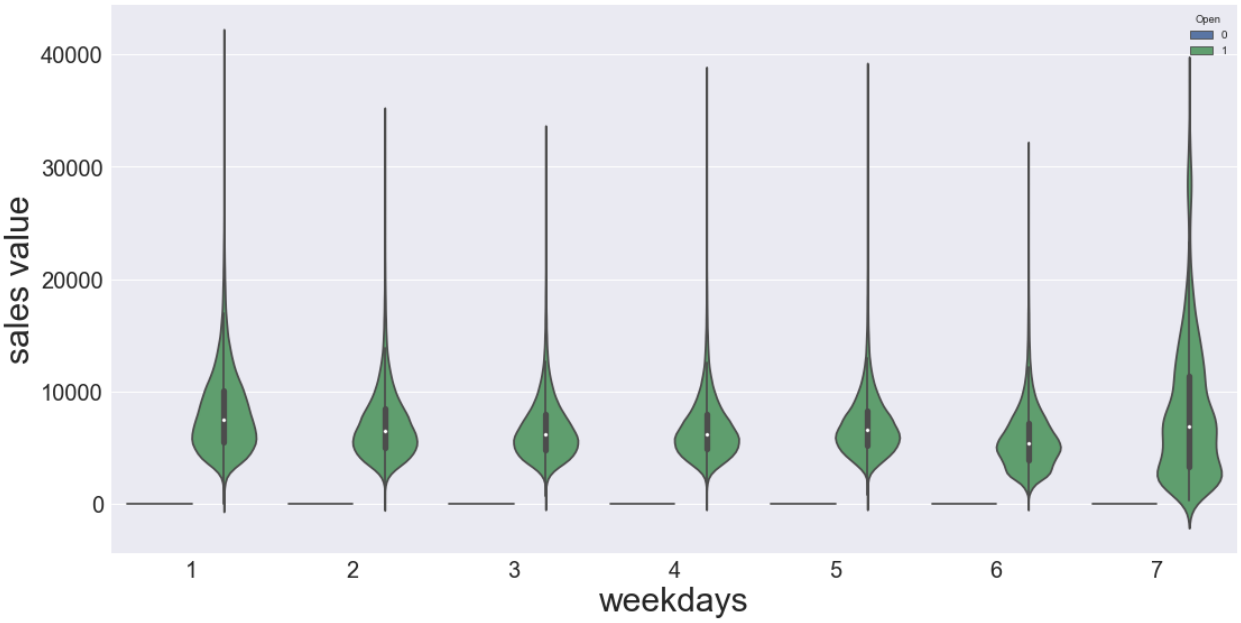

图 4: 销售额对应工作日/开业与否的 Violinplot

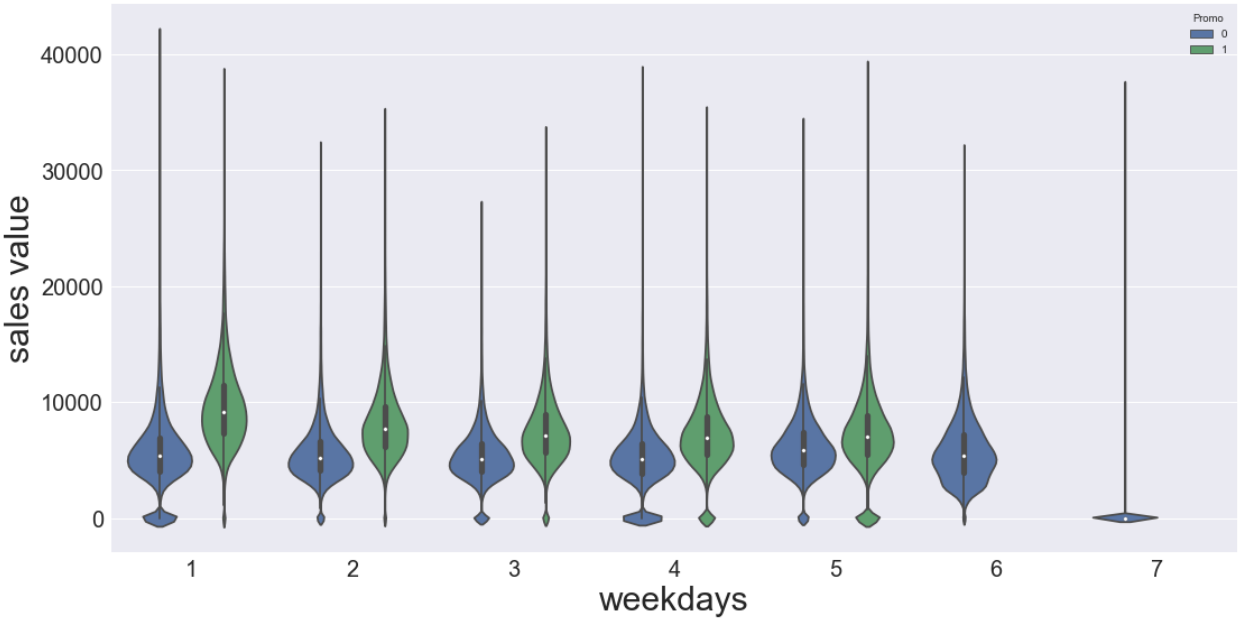

图 5: 在不同工作日，促销与否对应销售额的 Violinplot

果的一个重要的参数。在图5中，我们还可以发现一个特别的现象，在周六和周日，商店均没有开展过促销活动。且周日的销售额非常低，这也是符合逻辑的，因为在德国，绝大多数情况下 (除了节假日前后)，绝大部分商店-包括 Rossmann 都是不营业的，因此总的销售额会远低于工作日以及周六。

接下来，再分析一下销售额与其他特征之间的关系。销售额与国家节假日的类型可以由图6所显示。在这里可以明显的看出，非节假日的销售额要远大于节假日的销售额。而在三个不同种类的节假日类型下，虽然分布情况略有不同，但是整体的销售额均值差距不算大。

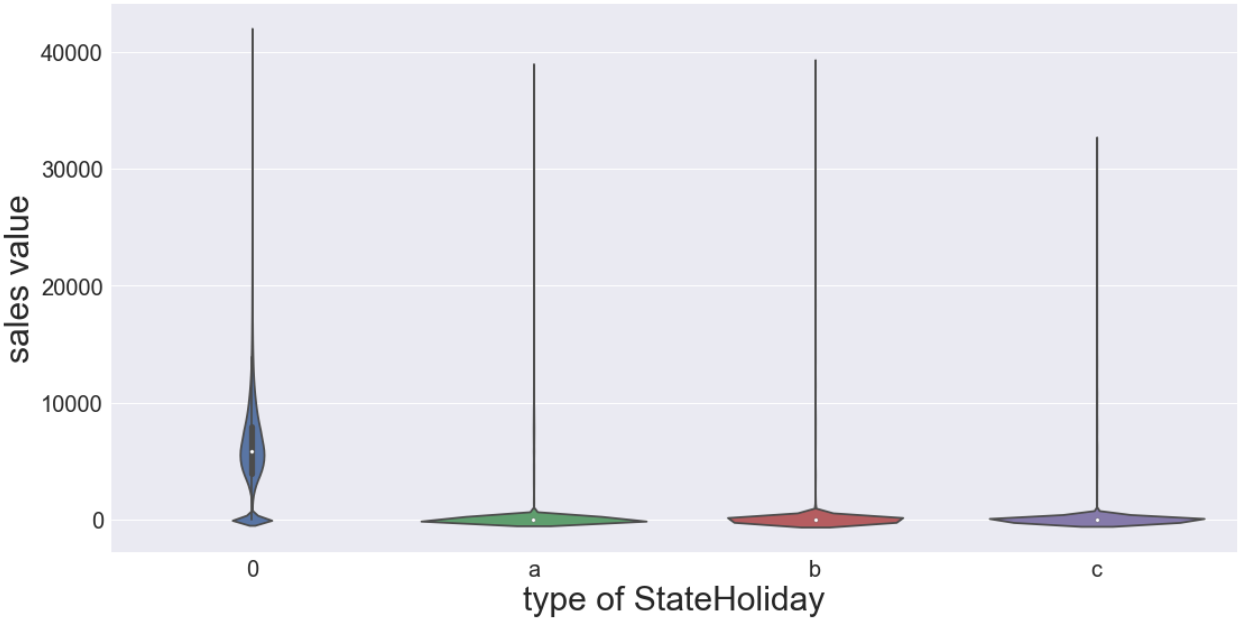

图 6: 销售额对应国家法定节假日的 Violinplot

销售额与是否为学校假期的关系可以由图7所表示。在这幅图中我们可以看出，学校放假与否对销售额的均值没有直观的影响。

将时间序列拆分成为年月日之后，首先分析一下在这不到三年的统计中，每个月的销售情况。这里使用的是 barplot，显示结果如图8所示。

从图8可以看出，商店的平均销售额和所对应的年和月有着显著的关系。2015 年的销售情况一共仅被统计了 7 个月。在 1，2，3，4，6 和 7 这几个月，2015 年的销售额均大于 2013 年和 2014 年。仅存在的一个例外是 2 月，在这个月里，2015 年的销售额超过 2013 年但是略低于 2014 年。如果将 2013 年和 2014 年的销售额对比的话，我们也可以发现类似的规律。也就是说在绝大多数的情况下，2014 年的销售额要大于 2013 年，存在例外的三个月是 3，7 和 8 这三个月。因此可以归纳出，Rossmann 商店的平均销售额在这三年里基本上是着年递加的。除此之外，还能被发现的一个比较明显的特性是，12 月份的销售额大于其他的月份。在这里可以推测出的原因是，因为 12 月份临近圣诞节和新年，可能会有不少人要送礼物；也可能是因为圣诞节新年的长假，另一部分人需要多准备囤积一些日用品在假期使用。所以，12 月份的销售额会比较明显的大于其他的月份。由图8可以看出，年 (Year) 和月 (Month) 这两个特征对销售额的影响还是比较大的。以上的这些分析可以由图9更加清楚的现实出来。在图9中，将 sales value 的区间范围缩小在 5250 到 7000 之间，则可以突出对比走势图曲线中不同的部分。

接下来再分析一下不同的日期 (DayofMonth) 对销售额的影响。统计的结果可以由图10所显示。从这幅

图 7: 销售额对应学校假期的 Violinplot

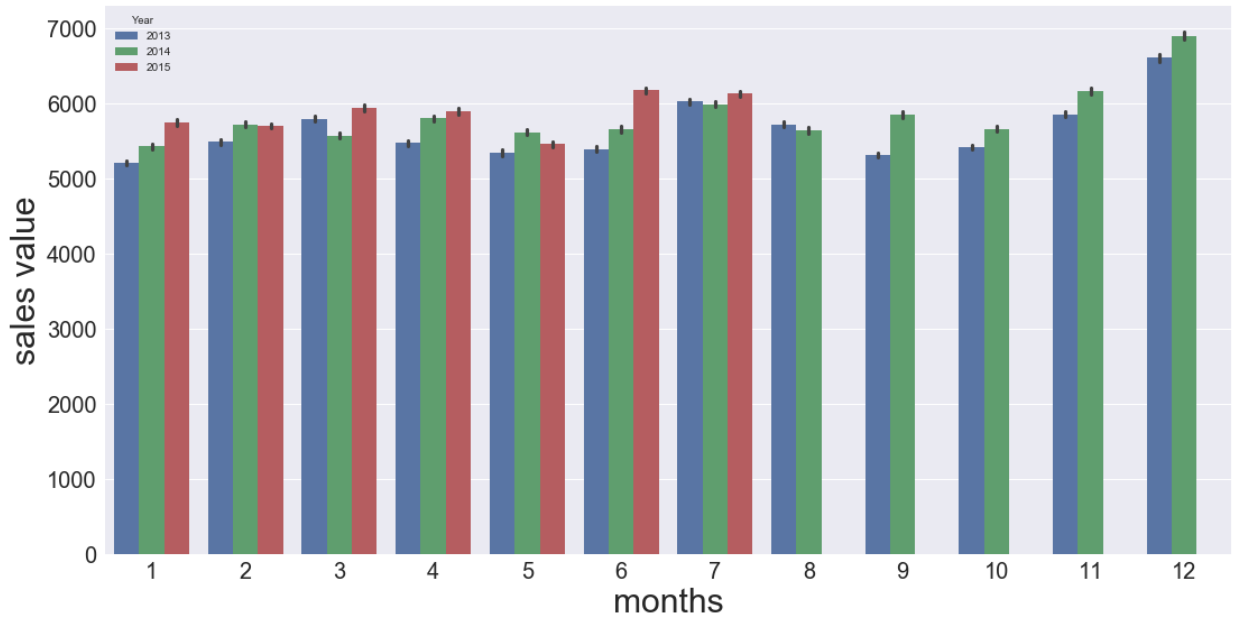

图 8: 商店的销售额对应不同的月份 (在这三年的统计中) 的 barplot

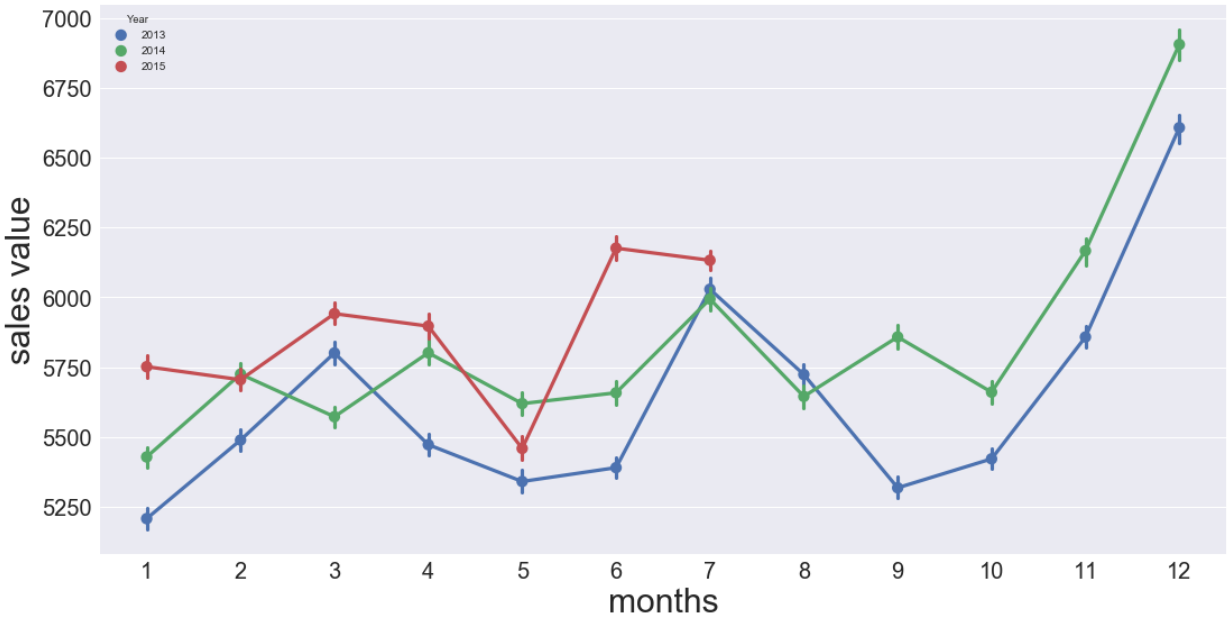

图 9: 商店的销售额对应不同的月份 (在这三年的统计中) 的 pointplot

图中，首先可以明显的发现，不同的日期对销售额的影响还是比较大的。不过这些影响并没有呈现出很由规律性的变化趋势。

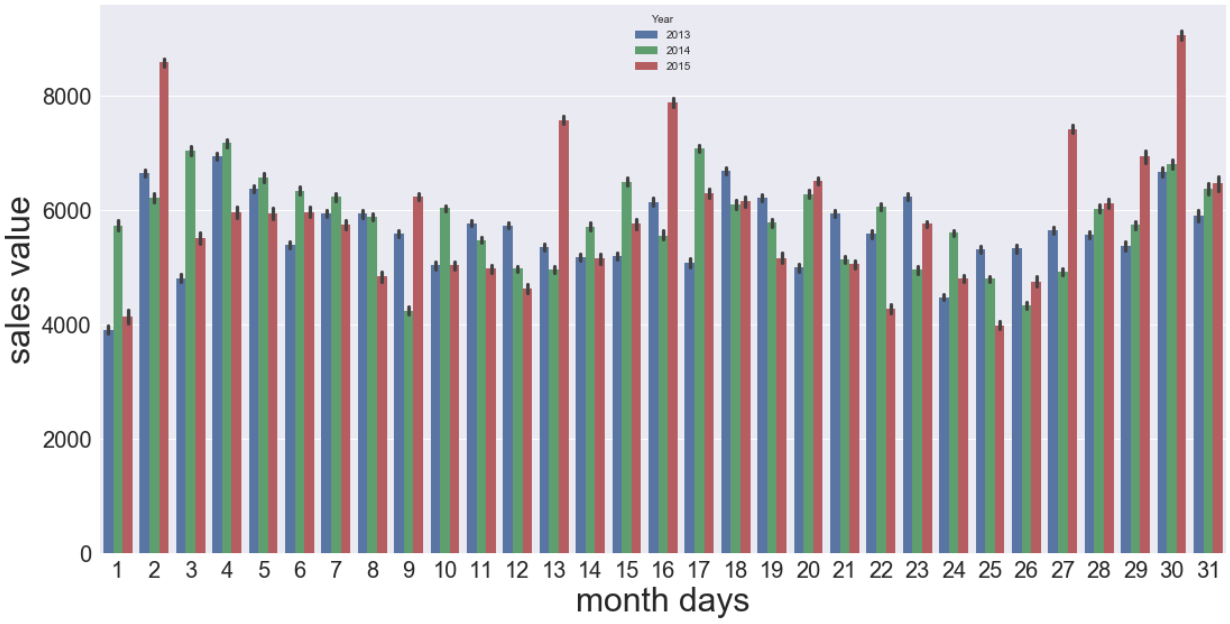

图 10: 商店的销售额对应不同的日期 (在这三年的统计中) 的 barplot

同样，也可以画出类似于图9的折线图，如图11所示。图11中可以可以看出，红色折线-对应的是 2015 年

的销售额-比其他两种曲线有着更大的波动。这是因为 2013 和 2014 年均统计了全年的数据，而 2015 年仅仅统计到了 7 月。按天来计算，统计的总次数相比前两年要小很多。所以才会出现这样不稳定的波动。除此之外，蓝色和绿色曲线的走势，也没有存在着非常明显的规律性特点。

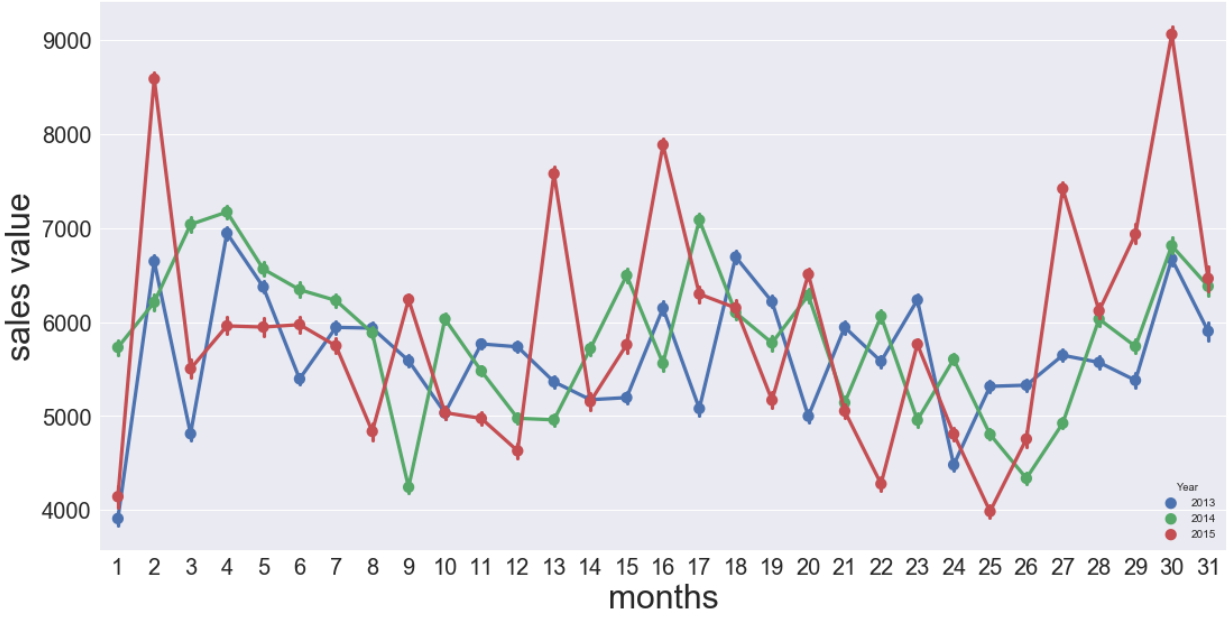

图 11: 商店的销售额对应不同的日期 (在这三年的统计中) 的 pointplot

由此，可以推测出的结论是，DayofMonth 会对销售额产生影响。但是这个影响是不是非常直接或者说，这个特征是不是一个非常重要的特征，需要在之后在之后的特征提取中，仔细分析讨论。

在这一章节中，我对”train.csv” 中的数据做了基本的可视化。主要分析了不同的特征对销售额的影响。Customer 这个特征被忽略掉的原因是，在”test.csv” 中并不存在这一特征。而从逻辑上舍去这一特征也是可以被理解的，因为在未来的某天造访商店的人数，应该是不会被提前知道的。

通过在这一章节里，对数据的探索性可视化分析，明确了数据集中存在着异常值，需要在之后的数据预处理的部分予以去除。此外还分析了在”train.csv” 中，不同的特征对销售额的影响，使得我们对数据集可以有一个比较直观的理解。而在上一个 Section 所提到的数据集合并问题 (将”store.csv” 合并至其他的两个数据集中)，也将会在之后的数据预处理章节予以具体讨论。

2.3 Algorithms and Techniques

在这个项目中，需要预测的数据是各商店的销售额，其他包含了很多不同的特征可与用于建立模型来预测商店的销售额。通过之前两个 Section 的分析，可以得出在数据集中存在不少的缺失数据，并且在数据集中有一些异常值需要剔除。除此之外，还涉及到数据集合并的问题。这些问题都是需要在开始用机器学习来建立模型之前需要解决的。因此，在这个项目中，首先需要被完整的步骤是：

- **特征工程及数据预处理**

首先，特征工程首先要解决的就是时间数据。时间数据是按照年-月-日 (Year-Month-Day) 的格式，但

是在机器学习中，这个格式的数据并不能直接被处理。可以考虑首先将其拆分，然后在对拆分后的数据进行处理。第二个需要处理的问题是需要将”store.csv” 和其他两个数据集”train.csv” 和”test.csv” 按照商店的 ID 进行合并。具体合并的方法会在之后的 Implementation 章节给以详细的说明。第三个是一些类别属性，比如分类为 a,b,c 的，可以考虑将其转化为数值型 1,2,3 进行处理。

除此之外，这个项目的数据集中有非常多的特征，不同的特征对销售额的影响也是不一样的。有些特征对预测的影响如果微乎其微的话，可以考虑将其去除，避免造成训练需要过长时间和造成维度灾难；而有些对销售额预测影响非常大的特征则一定需要保留。

接下来要做的则是一些特征组合 (因为特征组往往合可以是机器学习更有效率)，例如，训练集中给出的数据”Promo2SinceYear”(促销活动从哪年开始) 和”Promo2SinceWeek”。这个特征可以考虑和当天所在的年月日进行组合，得到促销活动一共持续了多少周，即”Prom-o2LastWeeks” 这样一个新特征。特征工程对我来说是一个比较新的概念，一些方法主要是参考了以下的网页[11]。

具体的特征特征工程的方法会在之后的数据预处理章节予以详细的说明和讨论。在完成了特征工程之后，在数据预处理方面还需要进行的是异常值的检测和去除。在之前两个 Section 中已经可以看出，数据集中的异常值是明显存在的。这些异常值将会影响机器学习的效率甚至是干扰到机器学习的结果。因此，在数据预处理的过程中，要将其找出并去除。

- **数据集拆分**

对数据及进行拆分，首先，在训练集中需要预留出一部分数据用作对模型的评价。使用 scikit-learn 中的 train_test_split 函数可以完成这一目标。在之前的章节，图1中可以看出，发现每个商店被统计的次数非常多，目测都有超过 9000 次。对数据集的拆分，一般来说会采用 80/20,90/10 或者 50/50 的比例 (Odom and Sharda, 1990)。举例来说，这里 80/20 指的是，将数据集的 80% 用作训练用数据，而将数据集的 20% 用作测试用数据。然后再对分出来的 80% 的数据继续分割出一部分用于 validation。当数据集非常庞大的情况下，可以提高这个比值；而当数据集相对较小的情况下，则可以减少这个比值来保证有足够的数据用于测试模型精度。

而在这个项目中，测试数据集”test.csv” 已经额外给出，所以可以将”train.csv” 中的数据全部都用来训练，换句话说，只需要将”train.csv” 中的数据分为训练集和验证集。因为每个商店被统计的次数均非常庞大，所以在这里，我打算仅仅使用 1% 的数据用作验证集，其余的 99% 的数据用作训练集。具体的分割方法会在 Implementation 的章节予以详细叙述。

- **Xgboost 算法**

首先，我通过对比赛第一名 Gent 赛后采访的详细阅读[12]，他所采用的 Xgboost 作为基准模型，并且使用了多个模型组合，最终获得了 0.10021 的高分。虽然在之前的课程中，我没有使用过 Xgboost 算法，但是打算在这个毕业项目中尝试使用这个算法来建立预测模型。

Xgboost 的全称是 eXtreme Gradient Boosting，属于 boosting 的一种算法，具体地说是基于 Gradient-Boosting 算法的一种优化版本，并越来越多的被工业界所使用[13]。

针对 Rossmann Sales 这么大的数据集来说，这种算法最大的优点是：速度快、效果好、能处理大规模数据等等[14]。首先考虑到使用 Xgboost 算法的原因是因为训练集的数据非常庞大。在”train.csv” 的数据集中，

[11] http://blog.csdn.net/dream_angel_z/article/details/49388733

[12] http://blog.kaggle.com/2015/12/21/rossmann-store-sales-winners-interview-1st-place-gert/

[13] http://blog.csdn.net/jasonzhangoo/article/details/73061060

[14] http://blog.csdn.net/a1b2c3d4123456/article/details/52849091

一共有 1,017,209 列数据。对于处理这么大规模的数据来说，Xgboost 是有优势的；其次考虑到训练数据包括各种不同属性的数据，通过正则化，可以控制模型的复杂度并且防止过拟合[15]。除此之外，Xgboost 还有很多一些其他的优势也是其他的监督学习算法所不具备或者不完全具备的。比如:Xgboost 支持并行处理和 CUDA 显卡加入计算[16]。此外 Xgboost 也是当今的数据科学家必备的工具之一，Kaggle 上的很多竞赛都离不开 Xgboost 算法[17]。除此之外，Xgboost 还有其他的一些有点，比如内置缺失值处理和交叉验证等。

因此，针对这个 Rossmann Store Sales 的项目，我决定尝试一下 Xgboost 作为解决方案。尽管参加 Kaggle 竞赛要想拿下好的排名的话，还需要运用到模型组合等其他的技能[18]，但是那需要更多的经验和时间。在这个项目中，我打算使用 Xgboost 作为基准模型，首先通过调节参数 (调参的方法具体参见如下网页所叙述 - https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python。) 看是否能够提升预测的精度；除此之外，可以考虑使用几次基础的，基于 Xgboost 算法的模型组合，采取求预测平均值的方法。优化一些过拟合的问题，从而提高预测的精度，取得更优的排名。

关于 Xgboost 算法的实现过程，具体会在 Implementation 章节给出详细的介绍和讨论。

2.4 Benchmark

这个项目是 Kaggle 上两年前的竞赛项目，一共有 3303 个组/个人参赛 (link:https://www.kaggle.com/c/rossmann-store-sales/leaderboard)。项目采用的评分标准是通过对比预测出来的销售额和真实的销售额之间的根均方误差百分比 (RMSPE)。这个值越小则会获得更高的评分，则会获得更高的排名。这个比赛的第一名 Gent，最终的评分是 0.10021。第 1000 名的得分是用户 juekoe，0.12109。这两个值，包括排行榜上的所有分数，以及对应的排名都是可以被查询到的，可以用作这个项目结果的参考数值。

按照 Kaggle 上的排行榜，可以看出自己所处的位置，我给自己的预期是先进入前 30%, 对应的分数是 0.12176。这个前 30% 可以作为一个基准，衡量自己的模型是否成功。接下来可以考虑通过调节 Xgboost 的参数来提高这个分数，并且可以考虑简单的求平均的方法，对几个 Xgboost 得到的模型进行组合 (因为对模型组合没有什么太多经验，可以考虑采用简单的求平均的方法)，重新提交进行评分。希望最终能进入前 15%，对应的分数是 0.11991。我预期最终排名很难进入前 10%(对应的评分是 0.11774)，因为特征工程对最终的模型评分起着很重要的作用，在这方面我还是处于刚刚开始学习的阶段，很多技巧和细节都需要经验; 同时，模型组合也是对于我来说也是一个很新的概念。但在最终的项目中，我会使用基准模型，通过调参和模型组合，尝试提高这个评分。我给自己定的目标是，希望可以进入前 15%。

换句话说，我给自己定下的目标是最低获得前 30% 的排名，在这个排名下，基本可以证明所建立的模型成功的完成了对商店销售额的预测 (虽然可能精度还有提升的空间)；如果最终排名能进入前 15% 的话，则表明这个预测模型还是很不错的。因为机器学习算法并不一定是这个项目唯一的核心，除此之外，对数据的处理 (特征工程) 是另一个核心的因素。只有把两方面都处理好，才能取得更高的排名。因此，我觉得，作为一个初学者，在这个比赛中如果能取得前 15% 的排名的话，可以算是一个很不错的结果。

[15] http://blog.csdn.net/Bryan__/article/details/52056112

[16] https://github.com/dmlc/Xgboost

[17] http://www.52cs.org/?p=429

[18] https://www.zhihu.com/question/29679189

3 Methodology

3.1 Data Processing

3.1.1 检测并去除异常值

在数据预处理章节，首先我们从去除异常值开始。在数据可视化的章节已经讨论过，该数据集存在着异常值。在这里，再次分析一下图1。可以发现这个 violinplot 的最上端和最下端的数据分布都是非常细长的，而数据的集中分布都在中段附近。这些非常细长的数据，特别是靠近端点的位置，则存在着异常值，需要予以剔除。消除异常值的方法有很多，比较广泛使用的 Turkey's test, 原理如图12所示[19]。

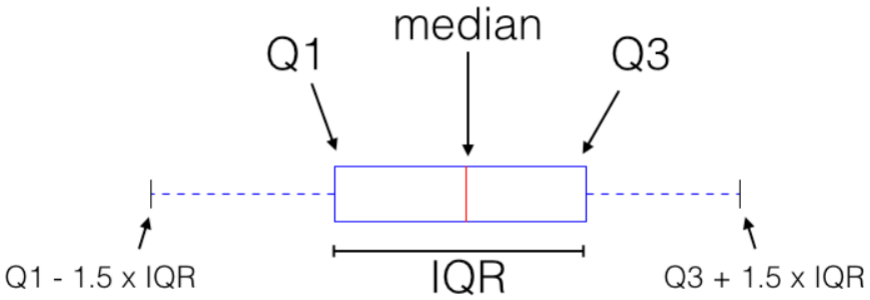

图 12: 使用 Turkey's test 去除异常值的原理图

在图12中，$Q1$ 指的是下四分位数，或者换句话说就是中间值到最小值之间的那个中间值。$Q3$ 是上四分位数，也就是中间值和最大值之间的中间值。IQR 指的是两个四分值之间的范围大小。在这幅图中，数值 1.5 是一个可变的系数 k，表示的是中度异常；对于重度异常的情况，则可以将系数 k 提高至 3。异常值的去除方法，以这幅图中出现的中度异常来看，所有比 $Q1 - 1.5 \times IQR$ 小的和所有比 $Q3 + 1.5 \times IQR$ 大的数值都应当被去除。在实际针对这个项目异常值处理的过程中，我针对了 k 取 1.5。具体的异常值去除的实施过程 (即如何在 python 程序中实现) 将在之后的 Implementation 章节中给以描述。这样，在原数据集中的 1017209 数据中，有 26694 被识别为异常值并予以去除。通过计算得知，被去除的数据一共占了总数的 2.6%。相对于数据的总量来说，这只是一个很小的数据集。剩下来的数据依然由足够的数量用来建立预测模型。应此，这个去除异常值的方法从这点看来，是可行的。

在去除了异常值之后，在重新绘制类似于图3的直方图和类似于图4的 violinplot。新的数据显示，由图13和图14所显示。

首先对比图3和13，发现在去除了异常值之后，数据分布变得“平滑”了很多，更加接近正态分布；再对比图4和14，通过对比发现在图4中的数据中明显的非常细长的曲线部分被去除了，而这部分数据中正是含有很多的异常值。图14中可以看出，数据分布的范围更加集中了。这两幅图 (图13和图14) 描述了去除异常值之后的数据分布情况。在完成了对异常值的检测和去除之后，接下来要进入的步骤是特征工程。

[19] http://www.grroups.com/blog/dixons-q-test-for-outlier-identification-a-questionable-practice

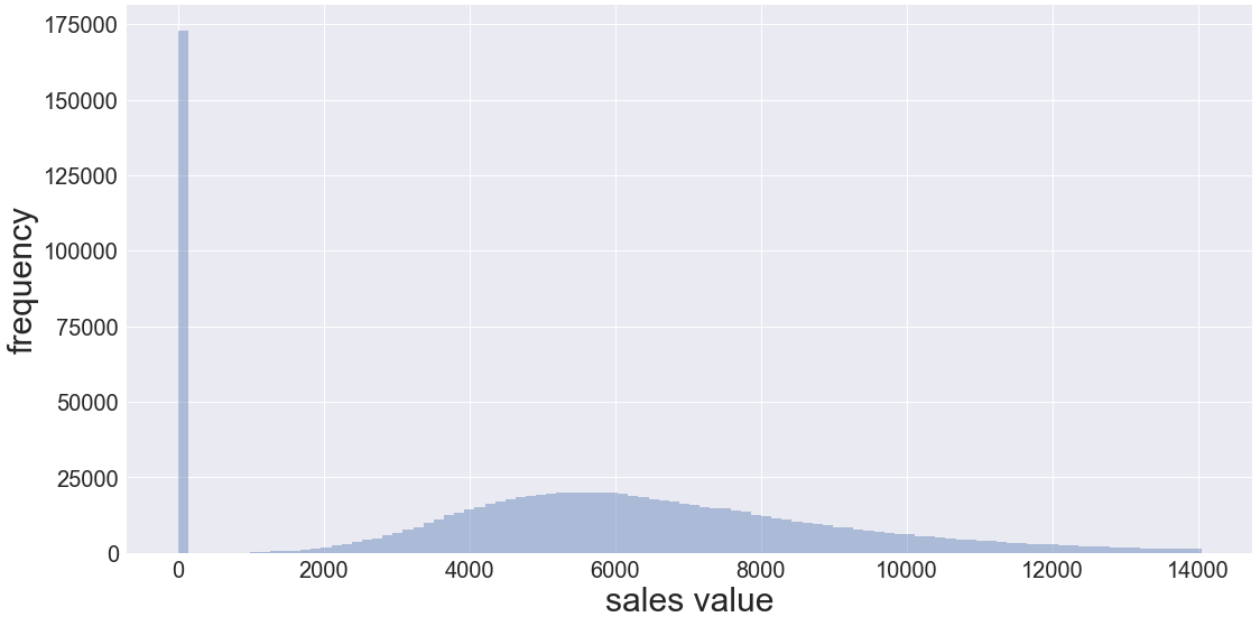

图 13: 所有商店销售额的直方图 (去除异常值之后)

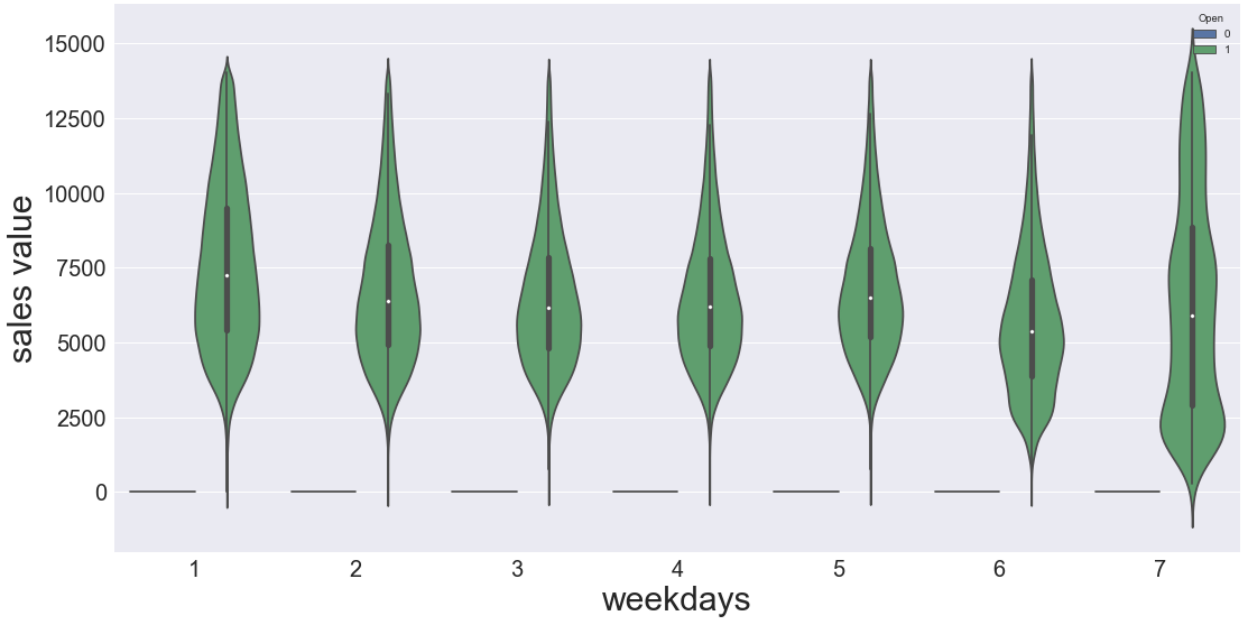

图 14: 销售额对应工作日/开业与否的 Violinplot(去除异常值之后)

3.1.2 特征工程

这个项目所用到的数据集中，需要用到大量的特征工程。接下来，我将按照最终的 Python 程序中的编码过程对特征工程和相关数据处理的方法按照步骤，给以说明。

- **1. 分离时间格式数据**

在数据集中所使用的时间格式是: $\%Y-\%m-\%d$(年-月-日) 的格式, 举个例子, 数据集中的第一个'Date' 特征表示为 $2015-07-31$。对机器学习来说，这样格式的数据不能直接被用作特征数据。在这个项目中，我对其进行了分离处理。我将这个格式的数据拆分成了以下几个子特征, 分别是:'Year', 'Month', 'DayOfMonth', 'DayOfYear' 和'WeekOfYear'。在这几个时间特征中，前三个是直观对应年-月-日的格式的。后两个'DayOfYear' 和'WeekOfYear' 是用作补充的两个特征。这样处理，关于特征的应用可能会存在一定量的重复问题以及增加了特征的维度。但是对于 Xgboost 来说，处理大维度的特征数还是可行的。所以在这里，先保留这五个特征，之后的分析中对影响较小的特征可以考虑予以去除。

在完成了以上步骤之后，还需要将'Date' 这个特征从数据集中去除，以免造成重复和无法被机器学习算法使用的问题。所以，在这里，经过处理后并保留下来时间日期特征分别是:

- 'Year'
- 'Month'
- 'DayOfMonth'
- 'DayOfYear'
- 'WeekOfYear'

- **2. 求销售额均值**

通过之前的分析可以看出，商店当天是否在搞促销活动，很大程度上的影响了商店的销售额情况。因此，首先，我对商店商店在搞促销活动和不在搞促销活动的情况下，分别计算了销售的均值。并分别对其命名为新的特征值'mean_store_sales_promo'(搞促销活动下的销售均值) 和'mean_store_sales_not_promo'(不搞促销活动下的销售均值)。

接下来我又分别计算了年销售均值 ('mean_store_sales_2013-2015'),三年间平均每个月的均值 ('mean_store_sales_m1-12'), 平均每个工作日的均值 ('mean_store_sales_d1-7') 以及商店前一个月 ('mean_store_sales_1month'), 两个月, 三个月和六个月
('mean_store_sales_2,3,6months') 的销售额均值。这样大量的均值明显是浪费了一些计算资源, 在之后 Xgboost 运行后，通过绘制特征值重点排序 (Feature importantce) 的柱状图，考虑仅仅保留如下的均值作为机器学习的特征值:

- 'mean_store_sales_promo'
- 'mean_store_sales_not_promo'
- 'mean_store_sales_2013'

- 'mean_store_sales_2014'

- 'mean_store_sales_2015'

- 'mean_store_sales_1month'

- 'mean_store_sales_2months'

- 'mean_store_sales_3months'

- 'mean_store_sales_6months'

- **3. Mapping**

在原始数据集中，我们可以发现很多数据都是以字母的形式所表现的。这些字幕需要被转化成数字才能够被机器学习的算法所利用。所以，Mapping(数值对应转换) 则是特征工程里需要处理的。

首先将特征'StateHoliday','StoreType' 和'Assortment' 中对应的 a, b, c, d 转换成数字 $1, 2, 3, 4$。接下来，将'PromoInterval' 中对应的'Jan,Apr,Jul,Oct','Feb,May,Aug,Nov' 以及-'Mar,Jun,Sept,Dec' 分别赋予数值 $1, 2, 3$。

- **4. 数据集合并**

使用 pandas, 将数据集'store.csv' 分别和数据集'train.csv','test.csv' 合并。合并的规则是按照商店的编号。合并完成之后，'store.csv' 中的所有特征均成为了训练集和测试集的特征。

- **5. 特征组合**

通过观察发现,以下四个特征'CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear','Promo2SinceWeek' 和'Promo2SinceYear' 从直观上看，和商店的销售额并没有直接的关系。首先我们知道了附近最近的竞争对手是从哪年哪月开始竞争的，由此可以计算出竞争持续了多少个月。即, 通过'CompetitionOpenSinceMonth' 和'CompetitionOpenSinceYear' 这两个特征，可以组合成新的特征'CompetitionLastMonths'。同理，知道了促销是从哪年，哪一周开始的话，则可以计算出促销从开始第一天到统计当天所存在的天数。即，通过'Promo2SinceWeek' 和'Promo2SinceYear' 来建立新的特征'Promo2LastDays'。

除此之外, 通过之前的分析可以看出, 商店的销售额在和 12 月的时候, 是明显高于其他月份的 (如图8和图9所示)。从直觉上推断，是因为圣诞节和新年将至，有不少人要给亲朋好友买礼物；也或许是有部分人要适当囤积一部分的日用品货物以备在假期中使用。不管原因如何，可以分析出，邻近年底的时候销售额大幅提高。由此可以推测出, 临近国家法定假日的情况下, 商店的销售额会受此影响, 相对平时有一定的变化。而这个特征，也就是距离法定节假日的天数，需要被重新提取出来。可以考虑使用以下两个特征'StateHoliday' 和'DayOfYear' 来组合出新的特征'DaysToHoliday'。

在这个步骤里，使用特征组合得到的几个新的特征分别是:

- 'CompetitionLastMonths'

- 'Promo2LastDays'

- 'DaysToHoliday'

这里，我仅仅对以上几个特征进行了组合并且得到了新的特征。当然这不是一个全面和最优的解决方案。因为特征工程方面，在这门课里并没有过多的涉及，目前我还属于非常初级的阶段。如何优化特征工程，则是在以后的工作学习中需要涉及的。

- **6. 缺失值处理**

使用 pandas 导入'test.csv', 发现这是一个 41088×8 的 DataFrame。在这个数据集中，相对于其他统计的数据，'Open' 设个数据一共有 11 个缺失值，数量为 41077。

对于这个数据集来说，有很多的缺失统计数据。CompetitionDistance 缺失了 3 个商店的数据；CompetitionOpenSinceMonth 和 CompetitionOpenSinceYear 缺失了 354 个商店的数据；Promo2SinceWeek, Promo2SinceYear 和 PromoInterval 缺失了 544 个商店的数据。

一般来说，处理缺失的方法是将缺失的数据都重新赋值为 0；也可以将缺失的数据整行删除。但是，因为 Xgboost 会自动处理缺失值[20]，并且按照作者的意思，Xgboost 会自动选择处理缺失值的方法。所以，在这个项目中，我仅仅是分析了缺失数据，但是并没有对缺失数据做进一步处理。一切都交给了 Xgboost 来完成。事实上，结果证明，Xgboost 对缺失值的处理还是值得信赖的。

3.2 Implementation

在这个章节，整个项目是如何被实现的将会按照步骤进行详细的描述，其中包括了数据读取，处理，算法等等是如何编写的等等。接下来，按照项目的流程，各个步骤将会包括代码将会被介绍。按照项目要求，代码全部都是使用 Python 以及相关的库来完成的。

- **1. 载入需要用到的库**

除了 Python 基本的功能以外，该项目还是用了 pandas, numpy, matplotlib 以及 Xgboost 的库。在这里，尤其要注意 Xgboost 使用的是 GPU 版本，安装过程由以下链接详细叙述[21]以及[22]。Xgboost 的库则由则收录在以下 Github 链接[23]。

```
import pandas as pd
import numpy as np
import xgboost as xgb
import datetime as dt
from sklearn.cross_validation import train_test_split
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use("Agg")
```

- **2. 读取数据集**

```
df_train = pd.read_csv("../input/train.csv", low_memory=False)
df_test = pd.read_csv("../input/test.csv", low_memory=False)
df_store = pd.read_csv("../input/store.csv", low_memory=False)
```

[20] https://github.com/dmlc/xgboost/issues/21

[21] http://www.picnet.com.au/blogs/guido/post/2016/09/22/how-to-build-xgboost-on-windows/

[22] https://github.com/dmlc/xgboost/blob/master/doc/build.md

[23] https://github.com/dmlc/xgboost

- **3. 去除异常值**

关于去除异常值的实施，主要是使用之前的数据预处理章节中所提到 Turkey's test 的方法。这里赋值参数 k(IQR 的系数) 为 1.5。除此之外，我还尝试了手动去除异常值，并编写了相应的代码，如下面所提到的方法 2 所示。为了保留更多的数据，我将对 Sales 统计的上限设置在 0.95，下限设置在了 0.001。在方法 1 中的参数 k，以及方法 2 中的参数 low 和 high, 他们的值都是可变的。

方法 1：按照 Turkey's test 方法编写

```
#     k = 3
#     k = 2.8
#     k = 2.5
#     k = 2
    k = 1.5
def remove_outliers(data):
    df_0 = data.loc[data.Sales ==0]
    median = np.median(data.Sales)
    q1 = np.percentile(data.Sales, 25, axis=0)
    q3 = np.percentile(data.Sales, 75, axis=0)
    iqr = q3 - q1
    df_temp = data.loc[data.Sales > q1 - k*iqr]
    df_temp = data.loc[data.Sales < q3 + k*iqr]
    result = df_temp
    return result
```

方法 2：手动筛选去除

```
low = .001
high = .95
def remove_outliers(data):
    df_0 = data.loc[data.Sales == 0]
    df_temp = data.loc[data.Sales <= high * np.max(data.Sales)]
    df_temp = df_temp.loc[data.Sales >= low * np.max(data.Sales)]
    frames = [df_temp, df_0]
    result = pd.concat(frames)
    return result
```

- **4. 分离年月日格式的数据**

```
time_format = '%Y-%m-%d'
def seperate_date(data):
    # 分离日期格式数据
    data_time = pd.to_datetime(data.Date, format=time_format)
    data['Year']= data_time.dt.year
    data['Month'] = data_time.dt.month
    data['DayOfYear'] = data_time.dt.dayofyear
    data['DayOfMonth'] = data_time.dt.day
    data['WeekOfYear'] = data_time.dt.week
    # 去除纯日期格式数据，保留处理过的
    data = data.drop('Date', axis=1)
    return data
```

- **5. 求一些销售额的平均值**

这段代码较长，就没有在报告中显示。详情请参见 main.py 文件中的'add_mean_sales' 函数。

- **6. 删除无用的商店 ID 信息**

通过分析，发现在'test.csv' 的数据集中，并不是让我们测试所有的 1115 家的商店。有部分的商店没有被列出。因此，我打算将这部分商店从训练集中移除，于是写下了以下代码。

```
def drop_stores(data_test, data):
    stores = data_test.Store.unique()
    for store in stores:
        serie = data[data.Store == store]
        data = serie
    return data
```

- **7. 补充特征工程**

这部分代码分别解决了：将'store.csv' 转成的 dataframe 和其他的 dataframe 按照商店序号合并；并且按照之前提到的特征组合建立新的特征'CompetitionLastMonths' 和'Promo2LastDays'；之后再接着进行 mapping，将字母特征转为数字特征。这段代码如下：

```
def feature_eng_compl(data):
    # 合并store入其他数据
    data = data.join(df_store, on='Store', rsuffix='_')
    data = data.drop('Store_',axis=1)
    # 特征组合，建立新的特征
    data['CompetitionLastMonths'] = 12 * (data['Year'] - data['CompetitionOpenSinceYear'].apply(
    lambda x: x if x > 0 else np.nan) - 1 + data['CompetitionOpenSinceMonth'].apply(lambda x: x
    if x > 0 else np.nan))
    data['Promo2LastDays'] = 365 * (data['Year'] - data['Promo2SinceYear'].apply(lambda x: x if x
     > 0 else np.nan))/4.0 + (data['DayOfYear'] - 7*(data['Promo2SinceWeek'].apply(lambda x: x if
     x > 0 else np.nan)) - 1)
    data = data.drop(['CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear', 'Promo2SinceWeek',
     'Promo2SinceYear'], axis=1)
    # mapping
    data['Year'] = data['Year'].map({2013:1, 2014:2, 2015:3})
    data['StateHoliday'] = data['StateHoliday'].map({'0':0, 'a':1, 'b':2, 'c':3})
    data['StoreType'] = data['StoreType'].map({'0':0, 'a':1, 'b':2, 'c':3, 'd':4})
    data['Assortment'] = data['Assortment'].map({'0':0, 'a':1, 'b':2, 'c':3})
    data['PromoInterval'] = data['PromoInterval'].map({'0':0,'Jan,Apr,Jul,Oct':1,'Feb,May,Aug,Nov
    ':2,'Mar,Jun,Sept,Dec':3})
```

- **8. 调用函数**

在这里，调用函数，依次运行如下步骤：异常值去除，分离时间格式，添加销售额平均值特征，和补充特征工程函数。这部分代码如下：

```
# 去除异常值
print ('remove outliers ..............')
df_train = remove_outliers(df_train)
# 分离时间格式
print ('seperate date ..............')
df_train = seperate_date(df_train)
df_test = seperate_date(df_test)
# 添加销售额平均值为新特征
df_store = add_mean_sales(df_train, df_store)
print ('add mean sales ..............')
# 补充特征工程
print ('feature engineering ..............')
df_train = feature_eng_compl(df_train).drop('Customers', axis=1)
df_test = feature_eng_compl(df_test)
```

- **9. 添加新特征'DaysToHoliday'**

```
holidaysofyear = df_train[(df_train['StateHoliday'] == 1)].DayOfYear.reset_index(name='
    DayOfHoliday').DayOfHoliday.unique()
holidaysofyear = sorted(holidaysofyear)
    for holiday in holidaysofyear:
    df_train['DaysToHoliday' + str(holiday)] = holiday - df_train['DayOfYear']
for holiday in holidaysofyear:
    df_test['DaysToHoliday' + str(holiday)] = holiday - df_test['DayOfYear']
```

- **10. 从数据集中去除以年月日为格式的'Date' 特征**

```
df_train = df_train.drop('Date', axis=1)
df_test = df_test.drop('Date', axis=1)
```

- **11. 定义 Xgboost 的评价函数，以及生成特征重要性图谱**

这部分代码是借鉴了 Kaggle 上的 Kernal，链接如下[24]:

```
# 建立feature_importance
def create_feature_map(features):
    outfile = open('xgb.fmap', 'w')
    for i, feat in enumerate(features):
        outfile.write('{0}\t{1}\tq\n'.format(i, feat))
    outfile.close()
# RMSPE计算公式
def rmspe(y, yhat):
    return np.sqrt(np.mean((yhat/y-1) ** 2))
# 在Xgboost算法中调用RMSPE作为评价函数
def rmspe_xg(yhat, y):
    y = np.expm1(y.get_label())
    yhat = np.expm1(yhat)
    return "rmspe", rmspe(y,yhat)
```

[24] https://www.kaggle.com/mmueller/liberty-mutual-group-property-inspection-prediction/xgb-feature-importance-python/code

- **12. 开始运行机器学习**

- **a. 定义训练集和测试集**

```
train = df_train
test = df_test
train = train[train["Open"] != 0]
train = train[train["Sales"] > 0]
```

- **b. 设置 Xgboost 的参数**

- 'objectiv'

首先需要将'objective' 这个参数设置为 reg:linear。这也是'objective' 这个参数的默认值。这样是将其作为线性回归的问题来处理。

- 'eta'

'eta' 这个参数在 Xgboost 中相当于学习率 (learning_rate)。默认值是 0.3，通过减少这个数值，可以提高预测模型的 Robustness。典型的值是 0.01 到 0.2 之间。在这里，我将其设为 0.1，这个值也有优化的空间。

- 'min_child_weight'

这个参数的作用是避免过拟合。通过提高这个参数的数值，可以避免模型学习到局部的特殊样本，从而影响模型的 Robustness。这个参数的默认值是 1，在这里我将其设置为 50。

- 'booster'

'booster' 这个参数是用于选择迭代的模型。可以被赋予的值有'gbtree'(基于树的模型) 和'gblinear'(线性模型)。这里需要选择为'gbtree'。

- 'alpha'

这个值的默认值为 1，提高这个值，可以使得 Xgboost 的算法速度变得更快，在适用于很高维度的情况下。我适当修改了这个值，使数值提高到 2。

- 'gamma'

'gamma' 这个参数所指定的是节点分裂所需要的最小损失函数下降值。一般来说，这个参数的值越大，算法会越保守。这个值也是为了提高模型的 Robustness。在这里，我将其设置为 2，作为一个初始值。

- 'subsample'

'subsample' 这个参数的作用是控制每颗数，随机采样的比例。减小这个值可以使得算法变得个更加保守，从而来避免过拟合；但是这个值不能是非常小的，否则会造成欠你和。默认值是 1，我设定为 0.9。

- 'colsample_bytree'

'colsample_bytree' 这个参数是用来控制每棵树随机采样列数的占比，作用和'subsample' 类似。典型值是 0.5-1，在这里我设置其为 0.9。

- 'silent'

静默模式设置。设置为 1，开启静默模式，不会输出任何信息。这个参数不影响模型建立。

- 'seed'

随机数种子，通过设置此数值，可以复现随机数据的结果。随机取值就可以了，默认值是 0。

- 'tree_method'

'tree_method', 因为我是用的是 GPU 版本的 Xgboost, 为了大幅提高运算速度, 这里我将'tree_method' 赋值'gpu_hist'；同时，对应的'max_bin' 参数设置为 600。

- 'max_bin'

仅仅当'tree_method' 选择为'hist' 的情况下，才需要设置这一参数。对应的数值设置在上一行已经给以说明。

- 'max_depth'

这个数值所描述的是树的最大深度，其作用很大，是用来避免过拟合的。典型的数值是 3-10。因为不确定如何选取该值比较, 我在这里写了一个 for 循环，分别对 4-10 层的最大深度进行了设置。并在之后的运算中分别利用这六个不同的数值建立了模型，依次分析。

以上的叙述均是在仔细阅读"Complete Guide to Parameter Tuning in XGBoost (with codes in Python)[25]"之后给以的叙述，这部分的代码如下：

```
for iter in range(0,7):
    params = {'objective': 'reg:linear',
              'eta': 0.1,
              'min_child_weight': 50,
              'booster' : 'gbtree',
              'alpha': 2,
              'gamma': 2,
              'subsample': 0.9,
              'colsample_bytree': 0.9,
              'silent': 1,
              'seed': 1301,
              'tree_method': 'gpu_hist',
              'max_bin': 600,
              'max_depth': 12 - iter}
```

这里我所选择的参数按照文中的叙述是符合逻辑的。不一定是最优的参数，但是应该也是在理想的范围以内。文中提到用 grid-search 来分别优化所有重要的参数。在这里我仅仅是对'max_depth' 的选择进行了分析和优化选择。我也尝试使用了文中的方法使用了 grid-search，但是发现在如此多的特征的情况下，对

[25] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

Xgboost 做参数优化使用 grid-search 的效率很低。因为按照我现有优化方法，最终该项目也取得了还算理想的结果，因此在这里我没有继续深入去做参数优化。但是通过对'max_depth' 的分析和优化发现，参数的选取可以直接左右 Xgboost 的模型精度。

- **c. 开始训练并建立模型**

这部分代码是借鉴了 Kaggle 上的 Kernal, 链接如下[26]。我将'num_boost_round' 参数提高到了 5000，是为了进行更多的迭代，有可能会提高训练精度。

```
num_boost_round = 5000
features = list(train.drop('Sales', axis=1))
X_train, X_valid = train_test_split(train, test_size=0.01, random_state=1)
y_train = np.log1p(X_train.Sales)
y_valid = np.log1p(X_valid.Sales)
dtrain = xgb.DMatrix(X_train[features], y_train)
dvalid = xgb.DMatrix(X_valid[features], y_valid)

watchlist = [(dtrain, 'train'), (dvalid, 'eval')]
gbm = xgb.train(params, dtrain, num_boost_round, evals=watchlist, \
  early_stopping_rounds=200,
  feval=rmspe_xg,
  verbose_eval=True)

print("Validating")
yhat = gbm.predict(xgb.DMatrix(X_valid[features]))
error = rmspe(X_valid.Sales.values, np.expm1(yhat))
print('RMSPE: {:.6f}'.format(error))
```

- **d. 预测销售额并保存结果**

```
dtest = xgb.DMatrix(test[features])
test_probs = gbm.predict(dtest)

result = pd.DataFrame({"Id": test["Id"], 'Sales': np.expm1(test_probs)})
result.to_csv("XG_"+str(file_name)+'.csv', index=False)
```

- **e. 保存模型**

```
gbm.save_model(str(file_name)+'.model')
```

要注意的是，步骤 c 到 e 都是在之前步骤 b 中的 for 循环里面的，按照迭代，依次尝试不同的参数值赋值给'max_depth'。

- **13. 利用简单的模型组合求平均值**

在这里，我新建了一个名为'Aver.py' 的程序，简单的对之前做出预测的所有结果求取平均值。主要是为了进一步避免过拟合和欠拟合的情况。也就是简单的利用求平均的方法来做模型组合，将 7 组 Xgboost 训练出来的模型预测的结果求平均。代码如下:

[26] https://www.kaggle.com/cast42/xgboost-in-python-with-rmspe-v2

```
import pandas as pd
import numpy as np
test = pd.read_csv("../input/test.csv", low_memory=False)
a = pd.read_csv("XG_0.csv", low_memory=False).Sales
b = pd.read_csv("XG_1.csv", low_memory=False).Sales
c = pd.read_csv("XG_2.csv", low_memory=False).Sales
d = pd.read_csv("XG_3.csv", low_memory=False).Sales
e = pd.read_csv("XG_4.csv", low_memory=False).Sales
f = pd.read_csv("XG_5.csv", low_memory=False).Sales
g = pd.read_csv("XG_6.csv", low_memory=False).Sales
result=[]
for iter in range(0,len(a)):
    r_temp = np.mean([
                      [a[iter]],
                      [b[iter]],
                      [c[iter]],
                      [d[iter]],
                      [e[iter]],
                      [f[iter]],
                      ])
    result.append(r_temp)
result = pd.DataFrame({"Id": test["Id"], 'Sales': result})
result.to_csv("XG_Aver.csv", index=False)
```

- **15. 画图-feature importance**

这部分代码参考了以下网页[27]:

```
create_feature_map(features)
importance = gbm.get_fscore(fmap='xgb.fmap')
importance = sorted(importance.items(), key=operator.itemgetter(1))

df = pd.DataFrame(importance, columns=['feature', 'fscore'])
df['fscore'] = df['fscore'] / df['fscore'].sum()

featp = df.plot(kind='barh', x='feature', y='fscore', legend=False, figsize=(6, 10))
plt.title('XGBoost Feature Importance')
plt.xlabel('relative importance')
fig_featp = featp.get_figure()
fig_featp.savefig('feature_importance_xgb.png', bbox_inches='tight', pad_inches=1)
```

- **14. 使用 Jupyter Notebook 做数据 EDA 和数据可视化**

这部分代码以及说明在程序中有详细的说明。详情请见'EDA_and_Visualization_Notebook.ipynb' 这个文件。

[27] https://www.kaggle.com/mmueller/liberty-mutual-group-property-inspection-prediction/xgb-feature-importance-python/code

3.3 Refinement

在项目刚开始的时候，还没有考虑到特征工程的情况下，我想仅仅使用现有的特征不加处理来进行机器学习，发现这并不可行。首先有很多缺失的数据没有得到分析和处理; 其次数据中有很多的字符型数据都要转化成数字格式; 还有时间格式数据等等。因此，特地补充了特征工程相关的一些知识，对数据更加有效的进行了处理。虽然还有很多提升的空间，但是最起码的保证数据被处理到机器学习算法可以使用，并且得到一个还算不错的模型精度。因此，总体来说，在做数据处理和特征工程这个环节，作为一个初学者来说，结果还是可以接受的。

至于特征工程方面可以提升的地方，空间一定是有的。特别是在特征组合方面，我仅仅是计算并添加了距离各个法定假日的天数这一新的特征。应该还有很多的特征可以用来组合并且分析其显著性。比如说，再更加深层次的挖掘学校假日对销售额的影响等等。基于在开题报告中给自己设定的目标已经得以实现，所以在这个项目中，我就没有投入过多的精力在特征工程上了。如果要进一步提升机器学习的结果，进一步优化实现特征工程，是很有必要的。

在算法方面，使用的 Xgboost。虽然 Xgboost 可以处理很多组不同的特征，但是对这个数据集以及我后来又添加了不少特征，如果使用基于 CPU 版本的 Xgboost，那会非常慢速。因此在这个项目中，我是用了 GPU 版本的 Xgboost 算法。虽然安装过程相对比较复杂，中间需要用 visual studio 编译，但是还是受益很多的。运算速度相对 CPU 版本的 Xgboost 有了非常大的提升。这样让我可以花较短的时间尝试不同的参数选择，也不用太担心过多的特征会大幅增加计算的时间。

然后在参数选择方面，一开始是使用默认的参数，发现训练结果一直存在不少的过拟合现象。模型的精度也一直不让人满意。后来通过仔细阅读 (基于"Complete Guide to Parameter Tuning in XGBoost" 这篇文章，之前有提及相关网站链接) 各个参数所对应的意义，以及如何调参。大致上选择了一组还算是合理的参数。之后，重点优化'max_depth' 这个参数。接下来，又通过将 7 个模型预测的结果求平均值的方法，进一步减小了过拟合和欠拟合的问题。最终的结果还是完成了开题报告中设定的目标。具体关于结果的讨论将会在下一个章节具体讨论。

至于在算法方面还有可以提高的地方，首先我能想到的就是进一步的参数优化。通过这个项目，我发现了，对于 Xgboost 算法来说，不同的参数值起到的作用是非常大的。虽然，我按照文章推荐的做法手动优化了参数，但这决不是最优参数。在将来的工作中，可以多对参数优化做一些研究。如果在不考虑所需时间的情况下，可以考虑使用 grid-search 来做参数优化，从而进一步提高模型的精度。

最后，在最后做模型组合的时候，我仅仅是对 7 组模型预测出来的值求了简单的平均值，也不包括任何的加权平均。这仅仅是一个新手解决问题的方式。在今后的学习工作中，可以进一步加强特征组合的学习和应用。除此之外，这 7 组模型都是选取 Xgboost 算法来建立，不同之处仅仅为使用的'max_depth' 这个参数的赋值不同。如何设定不同的参数建立更好的模型，或者尝试选取其他算法，比如在以下博客http://mabrek.github.io/blog/kaggle-forecasting/提到的 glmnet 算法等，用作模型组合，将有可能收获更好的结果。

综上所述，如果要进一步提高预测的精度，我认为未来应该需要重点在特征工程和 Xgboost 的调参，以及模型组合上下更多功夫。

4 Results

4.1 Model Evaluation and Validation

在开题报告中已经说明，因为这是 Kaggle 上的一个竞赛，所以最终对模型的评价我是通过将预测出来的结果上传至 Kaggle 的官方网站，让其自动进行运算的。首先，在进行可视化之前，我将所有上传的记录网页打印成了 PDF 文件。这个文件名为'Rossmann_Submission_pdf' 将作为附件和最终项目报告一并上传。评审可以查阅我递交之后经过系统自动计算的结果。文件越在最前端，则越接近最近提交。除此之外，所有被提交的.csv 文件也都被打包以供审阅。上传的文件名，例如'XG_0.csv' 的意思均可以在源代码中详细解读，在这里就不一一说明。下面的补充说明，比如，对应的'outlier_k=3' 的意思是，使用去除异常值程序里，k 对应的参数是 3。接下来，我会以可视化的形式来解读这些上传的结果。

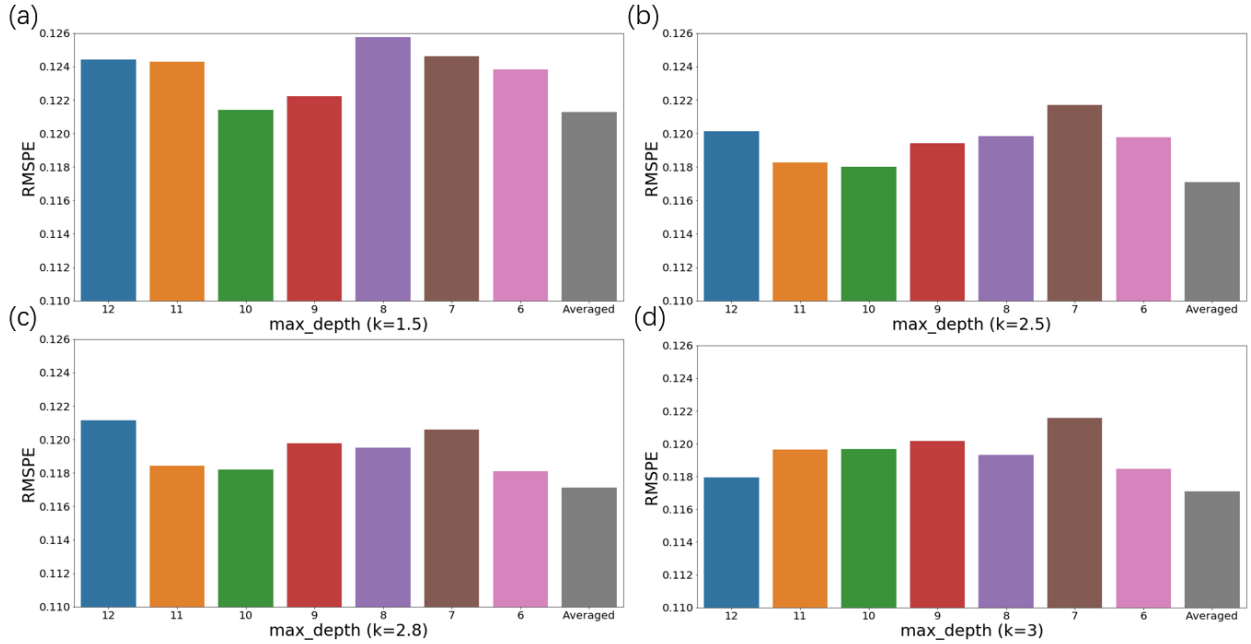

图 15: 按照不同的'max_depth' 和不同的异常值选取构建的模型的 private 评分

之所以选取 Private Leader Board 作为模型精度的评判标准是因为这个结果更为客观 (包含测试集 61% 的数据)。Public Leader Board 是在竞赛之间公开的数据结果，有 39% 的数据。参赛者可以根据反馈调整上传的数据结果，所以即使在 Public Leader Board 上获得很高的分数，也可能出现很多的 Overfitting 的现象，导致 Private Leader Board 的评分很低。

如图15所示，我首先将异常值去除的程序中，k 值分别设为 1.5,2.5,2.8 和 3。事实证明，异常值的筛选直接影响到了模型最终的精度。接下来，在每组设定好的 k 值下，对 Xgboost 里的'max_depth' 参数分别赋值 12 到 6 递减。接下来再对这 4 组 k 值下每组 7 个对应的'max_depth' 建立的模型的预测结果求均值。

从图15(a) 中可以看出，当 k 取 1.5 时，模型的精度 (RMSPE) 远远不如其他几组。通过对图15(b),(c) 和 (d) 的分析，首先可以看出一个规律：当'max_depth' 为 10 或者是 6 的时候，所建立的模型有相对较好的精度。如果是对这 7 组模型求平均值的话 (最简单的模型组合)，当 k=2.5 时，模型的 RMSPE 最小，值为:0.11708。

然后我就设想，如果将 (b),(c) 和 (d) 中所有模型所预测出来的值再求均值的话，会不会更大程度的降低过拟合和欠拟合呢？图16给出了答案。

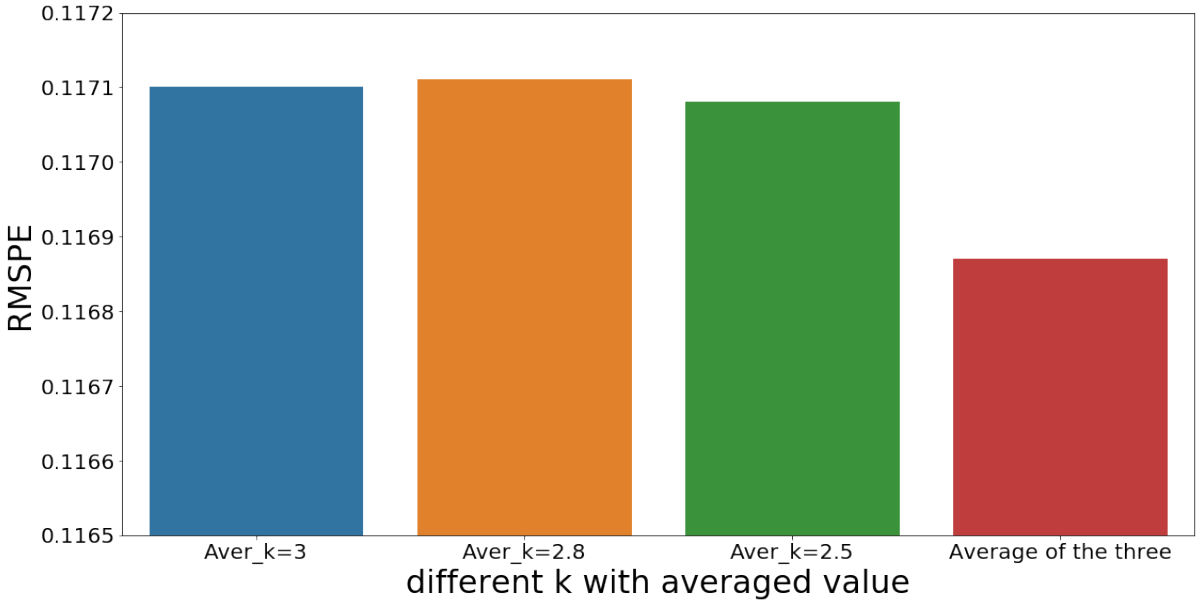

图 16: 所有模型预测结果均值对比单个 k 值对应的 7 组模型均值

在图16中，我们可以看出，所有模型预测结果的均值 ('Average of the three') 比其他三个均值的结果更好，值为:0.11687。对以上结果的讨论将在下节 Justification 中给以阐述。

4.2 Justification

首先，在建立的 28 个单个模型中，挑选出表现最好的模型是当 k=2.5,'max_depth'=10 的时候 (如图15(b) 所示)，此时 RMSPE 值为 0.11802。这个值对应的 Kaggle 上的排名是 351 名，对应的百分比是 10.6%。这个排名已经符合了在开题报告中所提出的单个排名进入前 30% 的目标。

接下来在分析之前通过求平均 (如图16) 所得到的最好的分数 0.11687。上传之后，发现这个值的排名是 253 名，对应的百分比是 7.6%。这个排名也符合了在开题报告中给自己定下的，通过简单的模型组合获得前 15% 排名的目标。

虽然结果是符合预期的，也证明了解决方案的合理。但是，项目的完成还有很多可以改进的空间以进一步提升模型的预测精度的。具体将在下一章节予以讨论。

5 Conclusion

5.1 Free-Form Visualization

关于模型的精度的可视化，已经在图15和图16中给出。在这里我打算补充绘制一幅关于特征重要性的图，如图17所示。

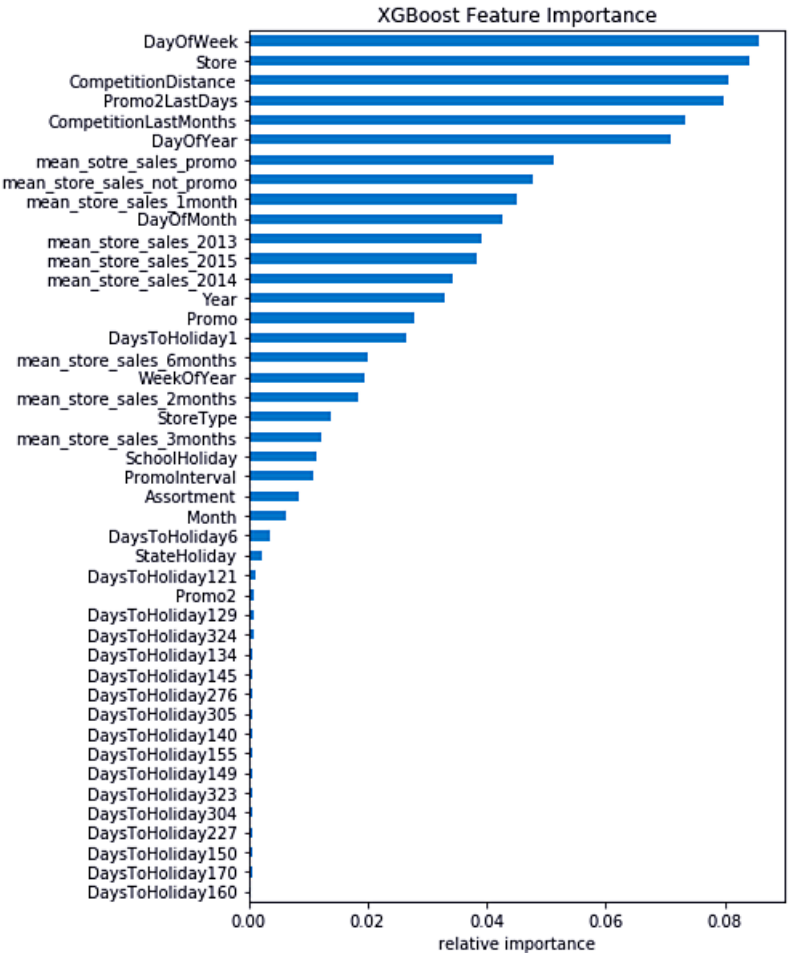

图 17: 特征重要性

首先需要说明的是，这幅图是在 k=2.5 的条件下绘制的。在图17中，我们可以清楚的看出各个特征重要性所占的比重。'DayOfWeek','Store' 和'CompetitionDistance' 都是对建立这个模型来说非常重要的特征。而最后几个特征'DaysToHoliday129' 到'DaysToHoliday160' 则是相对最不重要的几个特征。在建立模型方面所起的作用非常小。至于是否可以将这些参数删除以获取更高的运算效率，在这个项目中我没有去实施。因为我是用的是 Geforce GTX1070 的显卡运行 GPU 版本的 Xgboost 算法，运算速度还是非常快的。单组 5000 次跌代一般需要的时间不超过 10 分钟。所以在时间不是太重要的情况下，我选择了保留这些影响很低的特征。

从这幅图中我们还可以看出，过去 6 个月的销售额的均值对预测的影响大于过去两个月和三个月。而过去一个月的销售均值的影响是很大的，在特征重要性图的排序中排在前列。另外有一点比较好奇的是，2013 年销售均值对预测的影响大于 2015 年，这也是凭想象所无法预判的。在这幅图特征重要性图中，还有很多的有用信息，在这里我就不一一分析了。

5.2 Reflection

我对这个项目感兴趣的原因是，它处理的是一个基于真实数据集的常见的商业问题。我希望通过对这个项目的完成使得自己初步具有将机器学习运用在数据科学中的能力。

在项目开始的时候，自己还完全是一个数据分析方面的新手。于是在此期间，强化补充了 pandas 的知识用于处理分析数据集。另外还补充了 seaborn 的知识用于做数据可视化。这些东西都是在之前的课程中所没有涉及的，花费了不少的时间。

即便是这样，在做特征工程的时候，还是显得有点外行。一开始，使用默认的数据直接去进行机器学习，却甚至都无法跑起程序。经过了比较长时间的准备工作，才使得数据可以被算法所用。之前没有好好计算，但是花在数据预处理和特征工程方面的总时间占了项目完成所需时间的很大比例。

接下来在使用 Xgboost 算法的过程中，因为之前没有接触过，所以花费了一定的时间去查阅和学习这个算法。至于参数的选取，也是参阅网络上一篇推荐度比较高的文章，慢慢尝试和筛选出来的结果。

5.3 Improvement

我确信这个项目还是有很大的提升空间的，主要为以下几个方面:

- **特征工程**。

在这个项目中，我仅仅是对特征做了比较简单的处理。关于特征组合和特征提取方面，应该还有很长的路要走。而特征工程完成的好坏将直接影响最终的模型精度和可靠性。

- **算法参数选取**。

虽然在这个项目中，对参数的选取我是参考了"Complete Guide to Parameter Tuning in XGBoost" 这篇文章 (链接在上文中给出)。但是关于参数的选取还是手动完成的。只有对'max_depth' 这个参数进行了优化。而其他的参数理论上来说都存在被优化的空间。关于如何高效的使用 grid search 来优化 Xgboost 的参数，也是以后需要学习和强化的地方。

- **算法部分**。

在这个项目中，我仅仅是使用了 Xgboost 这个算法。虽然这个算法被很多的数据科学家推广和广泛被使用在工业界，但还有很多类似甚至更适用这个项目的算法有待发掘。

- **模型组合**。

在模型组合方面，我仅仅使用到了最基本的对模型预测结果求平均值的方法。除此之外还有很多更复杂的方法比如加权平均等等。

综上所述，项目的完成和优化还是存在空间的。新的技术则需要在以后的学习和工作中慢慢强化。

References

Jason Brownlee. Supervised and unsupervised machine learning algorithms. *Machine Learning Mastery*, 2016.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, pages 436–444, 2015.

Marcus D Odom and Ramesh Sharda. A neural network model for bankruptcy prediction. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 163–168. IEEE, 1990.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.