Predict Future Sales

方鑫杨 胡晨璐 梁孝转 周玉立

2019 年 1 月 10 日

1 介绍

这个挑战是Coursera课程“How to win a data science competition”的最终项目。

1.1 题目背景

在本次竞赛中，我们将使用由每日销售数据组成的具有挑战性的时间序列数据集，该数据集由俄罗斯最大的软件公司之一1C公司提供。1C公司成立于1991年，专门从事大众市场软件的开发，分销，发布和支持。而本次竞赛的数据，正是1C公司的部分商品在不同商店的销售记录。最终的目标是让我们预测每个商店不同商品的下个月的销售量。

1.2 相关工作

通常对于一个时间序列问题，人们通常会从时间的角度把一个序列分成三类，分别是：纯随机序列（白噪声序列）、平稳非白噪声序列和非平稳序列。通常来说，这类问题的思路往往是根据时序图和自相关图的特征做出主观的判断，而平稳的序列自相关图和偏自相关图不是拖尾就是截尾。如果是平稳序列，目前最常用的拟合平稳序列的模型为ARMA（Autoregressive moving average）模型 [1]，全称是自回归移动平均模型，

他又可以分为AR模型，MA模型和ARMA模型三大类。一般分析步骤如下：

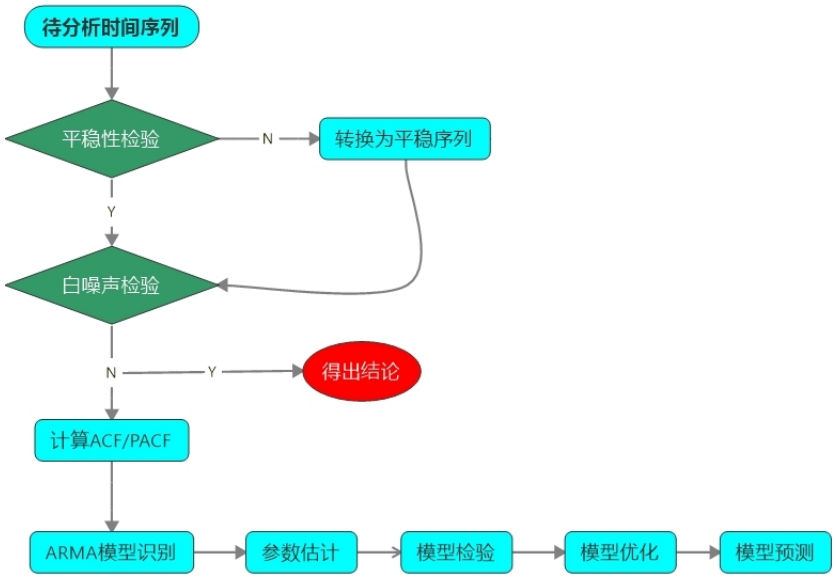

图 1: 平稳序列一般分析流程

如果是非平稳序列，我们需要先把它转为平稳序列之后再进行分析。一般我们使用ARIMA(Autoregressive Integrated Moving Average model)进行分析 [2]。但是以上只是常规的时间序列问题的一般解法流程。针对不同的问题，可能会有不一样的特点和更适合的特征，例如我们的问题，使用常规的分析方法则不能得到很好的效果。

2 数据描述

数据提供方一共提供了6份数据，其中sales_train.csv是训练集，test.csv测试集，items.csv是商品的数据集，item_categoties.csv是商品种类的数据集，shops.csv是商店的数据集，sample_submission.csv是提交的样本数据集。

2.1 sales_train

- date:按照日/月/年的格式的日期
- date_block_num：为了使用方便的连续月数。2013年1月为0,2013年2月为1,…， 2015年10月是33，要预测的为2015年11月，记作34
- shop_id：商店的唯一编码
- item_id：商品的唯一编码
- item_price：商品的价格
- item_cnt_day：售出产品数量。要预测的是这个度量的每月数量

2.2 items

- item_name：商品名称
- item_id：商品的唯一编码
- item_category_id：每种商品所属类的编码

2.3 shop

- shop_name：商店名称
- shop_id：商店的唯一编码

2.4 categories

- item_category_name：商品所属类的名称
- item_category_id：每种商品所属类的编码

2.5 test

- ID：测试集的唯一编码
- shop_id：商店的唯一编码
- item_id：商品的唯一编码

2.6 sample_submission

- ID：测试集的唯一编码
- item_cnt_month：要预测的第34个月售出的产品数量

3 特征工程

3.1 数据理解

在刚拿到数据时，我们首先对数据进行了一个大概的浏览如下：一共有2935849条销售记录

	date_block_num	shop_id	item_id	item_price	item_cnt_day
count	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06
mean	1.456991e+01	3.300173e+01	1.019723e+04	8.908532e+02	1.242641e+00
std	9.422988e+00	1.622697e+01	6.324297e+03	1.729800e+03	2.618834e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00	-2.200000e+01
25%	7.000000e+00	2.200000e+01	4.476000e+03	2.490000e+02	1.000000e+00
50%	1.400000e+01	3.100000e+01	9.343000e+03	3.990000e+02	1.000000e+00
75%	2.300000e+01	4.700000e+01	1.568400e+04	9.990000e+02	1.000000e+00
max	3.300000e+01	5.900000e+01	2.216900e+04	3.079800e+05	2.169000e+03

图 2: 数据描述

接着我们检查了一下异常值，我们绘制了item_cnt_day的箱式图，认为销量超过1000的离群值为异常值，接着绘制了特征item_price的箱式图，认定价格超过300000的商品也为异常值。在上图的我们发现有商品价格为负数的值，我们也将其作为异常值。

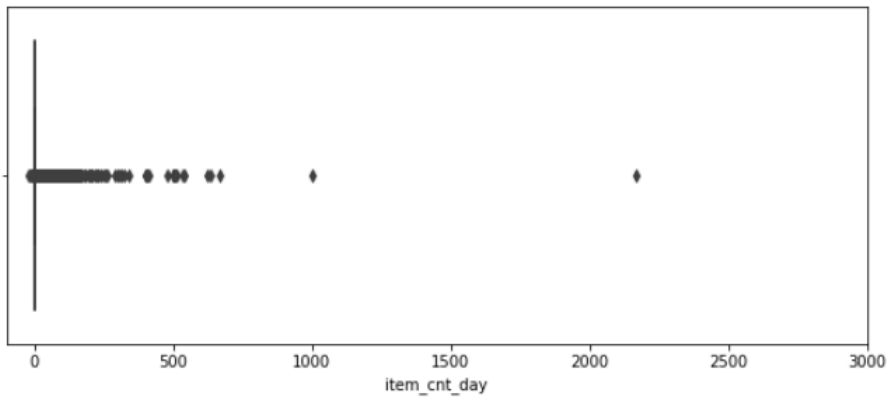

图 3: item_cnt_day箱式图

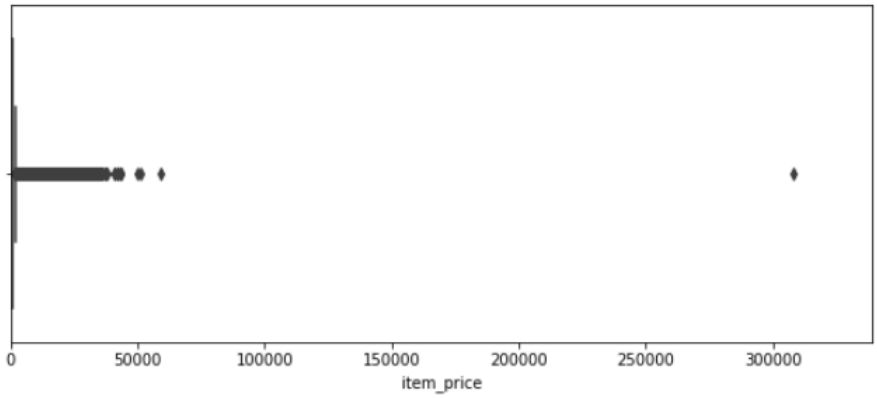

图 4: item_price箱式图

对于商店信息，统计过后一共有60个商店。对于商品信息，我们统计了共有22171个商品，对于商品的类别，共有85种。

3.2 数据清洗

针对检测出来的异常值，我们要对其进行处理。首先是离群值的处理，我们去除了商品价格大于100000的商品记录和商品销量大于1000的商品记录。对于商品价格小于0的商品记录，我们用商品价格的中位数去替换。并且通过检查，我们发现了有6个商店之间各有重名现象，我们对其进行了修改。

3.3 特征拆分

在shop_name中，前面是城市名，我们将其提取出来，单独作为特征city。商品类别中，在商店类别中，type中前面是商品种类，后面是商品具体的类别，将其划分为type和subtype两个特征。根据商品类别名，我们将商品划分为12个大类，具体为: PC Headsets / Headphones,Access, Tickets (figure), Delivery of goods, Consoles, Accessories for games, CD game, Card, Movie, Books, Music, Gifts, Soft, Office, Clean, Elements of a good。

3.4 特征聚合

测试集是预测每个商店每个商品的销售量，并且题目中提到了商店和产品每个月都会发生变化，所以我们对商店和商店的ID进行了笛卡儿积操作，形成了商品和商店之间的关联，每个商店和商品形成了一个pair。因为是预测各个商店下个月的销量，而train数据集中给出的特征是每天的销量，需要对其进行sum聚合，得到了item_cnt_month特征，即商品的每个月的销量。 Train数据集中有item_price和item_cnt_day特征，可以构建每日的收入特征revenue。在test数据集中加入date_block_num特征，设为默认值是一个月的值为34。最后将test数据集加入到train数据集中，并且将shops, cats, items数据集合并进来。

3.5 Lag生成特征

为了将时间序列问题转化成监督学习，需要做滞后处理，一条记录做训练会失去它几个月前的联系，就是破坏了时间序列的数据价值。因此我们对特征做了lag处理，将某一条数据对应的商店和商品，滞后了数个月，将某个商店的某件商品的上一个月的销量作为一个特征，因此一条记录就与前几个月的数据联系起来。我们定义了一个lag_feature函数，用于将时间序列数据集中的某些特征做滞后处理，生成新的特征。即将某些属性滞后，将前一时间步的值作为新的特征值。我们将item_cnt_month特征滞后1到12个月，产生12个新的特征，利用xgboost的回归模型，初步训练模型，得到新的特征的importance值，从中筛选出滞后了1、2、3、6、12个月的特征作为新的特征。

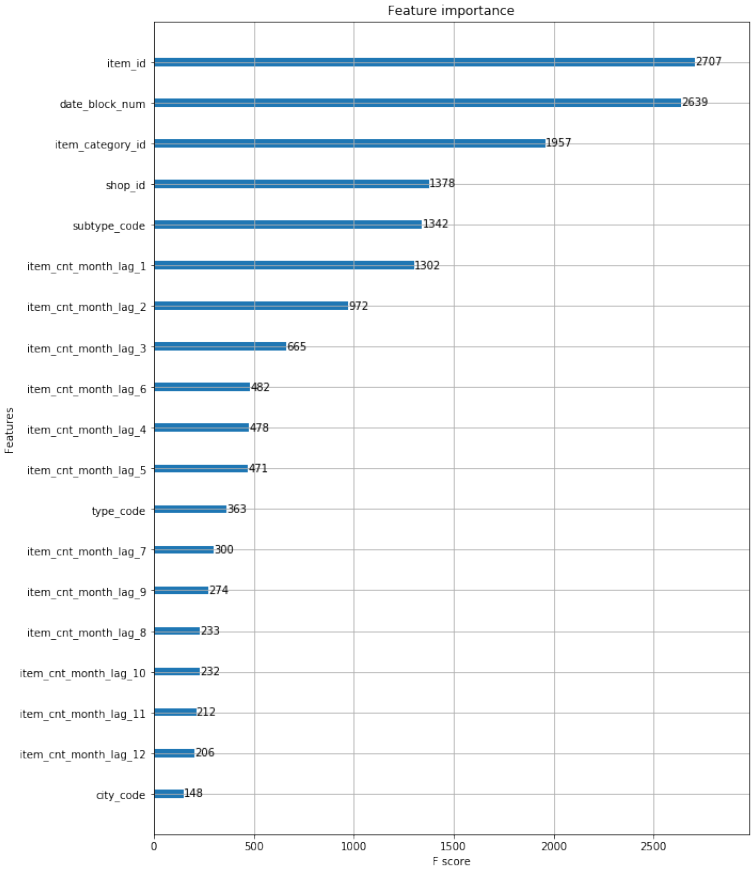

图 5: 特征重要性a

将item_cnt_month关于data_block_num, item_id分组，进行mean聚合，即得到每个月每个商品的平均销量，生成data_item_avg_item_cnt特征，再对新特征进行滞后，滞后步长分别设为1到12，再利用lightGBM的回归模型尝试训练，将得到的模型进行特征评估，选取了1、2、4时间步产生的新特征。

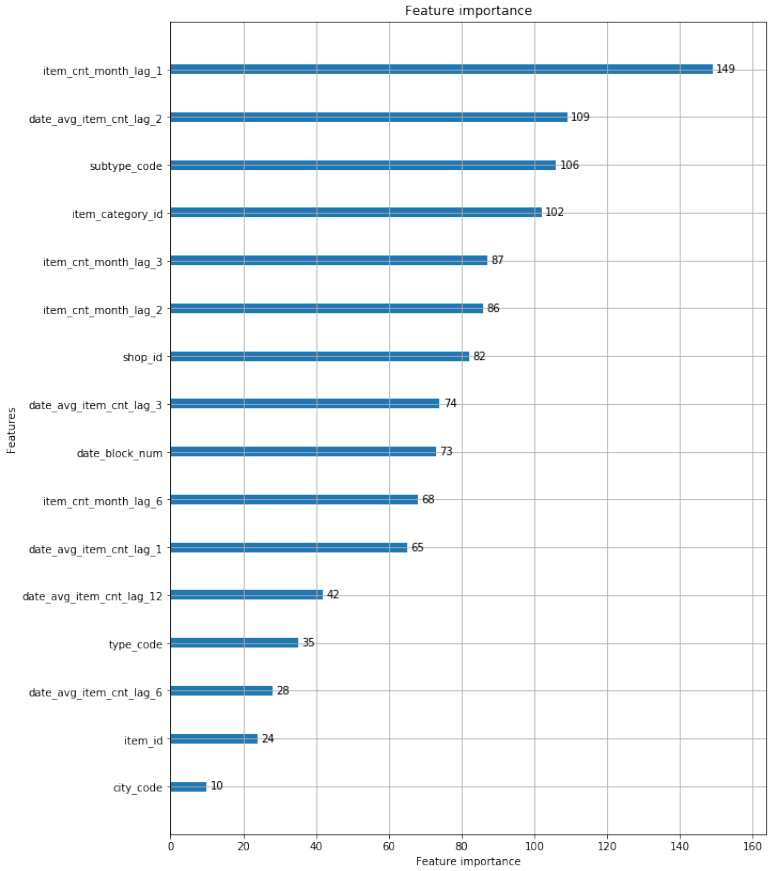

图 6: 特征重要性b

同上面操作，分别对shop_id, item_category_id, type_code, subtype_code,做与data_block_num相关的聚合操作，并对其进行滞后，筛选了其中importance较高的新特征。

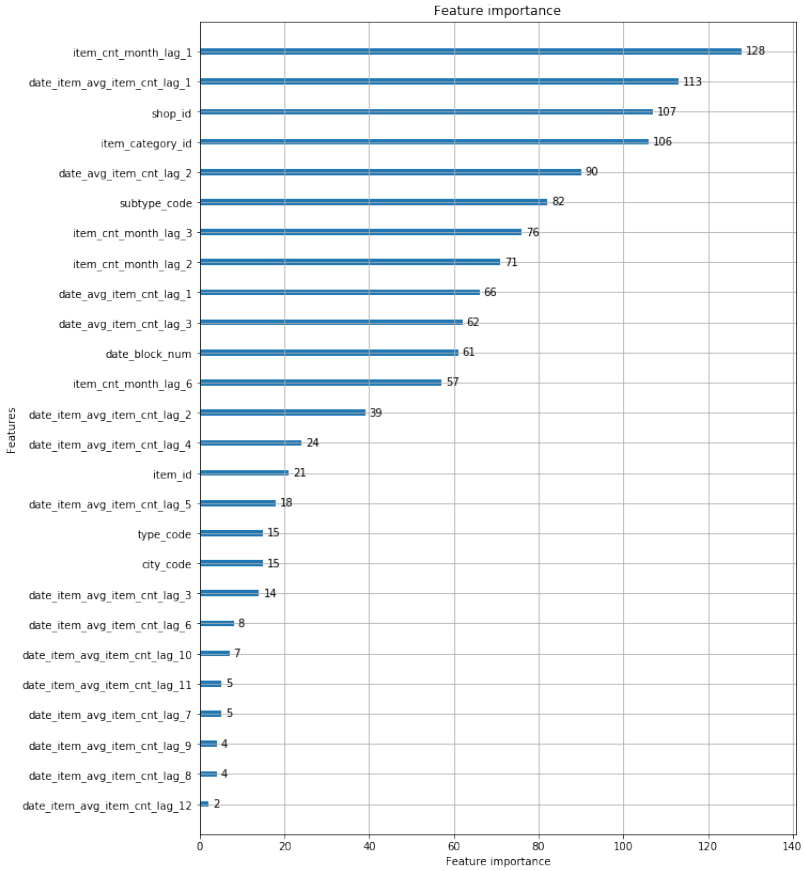

图 7: 特征重要性a

3.6 价格趋势

我们计算了商品的平均价格item_avg_item_price，商品每个月的商品价格date_item_avg_price，并对属性进行lag处理生成，滞后1到6个月的相关特征。并且计算商店每个月的总收入date_shop_revenue，以及商店的平均收入shop_avg_revenue，并做了lag处理。有的商店的有些商品可能是好几个月没有销售记录，我们计算了每件商品上一次销售时间距离当前的时间作为新的特征item_last_sale，以及每个商店每件商品上一次销售时间距离当前时间item_shop_last_sale。

4 模型选择

对于时间序列预测问题，常用的处理方式分为三个大类：传统时间序列处理方法、机器学习和深度学习。针对以上三种方法，我们分别采用了三种模型进行实验：Prophet [3]、LSTM（Long Short-term Memory） [4]和树模型。

4.1 Prophet

Prophet是Facebook发布的基于可分解（趋势+季节+节假日）模型的开源库。它让我们可以用简单直观的参数进行高精度的时间序列预测，并且支持自定义季节和节假日的影响。该模型可分解为三个主要组成部分：趋势，季节性和节假日。它们按如下公式组合：

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

其中：

- g(t)：用于拟合时间序列中的分段线性增长或逻辑增长等非周期变化
- s(t)：周期变化（如：每周/每年的季节性）
- h(t)：非规律性的节假日效应（用户造成）
- ε_t：误差项用来反映未在模型中体现的异常变动

我们在实验中使用的模型如下：

```
growth = 'linear'
m = Prophet(growth, yearly_seasonality=True)
```

图 8: Prophet参数

其中：

- growth：趋势参数
- linear：表示是线性的趋势
- yearly_seasonality：是一个季节参数，表示周期为年的季节性

4.2 LSTM

时间序列变量之间往往存在着依赖性，而循环神经网络（RNN） [5]正适用于处理序列依赖的问题，长短期记忆网络（LSTM）则是在RNN的基础上进行了改进。本实验基于keras深度学习框架，尝试了单层和双层的LSTM神经网络。实验结果表明，双层的LSTM优于单层的LSTM。
模型结构如下：

```
model = Sequential()
model.add(LSTM(64,dropout=0.2,recurrent_dropout=0.2,return_sequences=True))
model.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2,return_sequences=False))

model.add(Dense(1))

model.compile(loss='mse',optimizer='RMSprop',metrics=['mean_squared_error'])
```

图 9: LSTM参数

在这个双层LSTM中，第一层有64个神经元，第二层有128个神经元，最后用一个全连接层输出一个预测值。为防止过拟合，每个隐藏层有20%的数据被dropout掉。

4.3 GBDT

梯度提升决策树（GBDT） [6]是目前用于从结构化数据构建预测模型的最佳技术。GBDT利用前一个模型的误差，来建立下一个模型。此过程会依次减少模型的错误，因为每个连续模型都会改进先前模型的弱点。XGBoost [7]和LightGBM [8]框架拓展和改进了GBDT，实验结果表明，LightGBM在运行时间上相比XGBoost用一定优势。

```
model = XGBRegressor(
    max_depth=10,
    n_estimators=1000,
    min_child_weight=300,
    colsample_bytree=0.8,
    subsample=0.8,
    eta=0.3,
    seed=42)

model.fit(
    X_train,
    Y_train,
    eval_metric="rmse",
    eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
    verbose=True,
    early_stopping_rounds = 10)
```

图 10: XGBRegressor参数

```
model = LGBMRegressor(
    max_depth=8,
    n_estimators=1000,
    min_child_weight=300,
    colsample_bytree=0.8,
    subsample=0.8,
    eta=0.3,
    seed=42)

model.fit(
    X_train,
    Y_train,
    eval_metric="rmse",
    eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
    verbose=True,
    early_stopping_rounds = 10)
```

图 11: LGBMRegressor参数

在n_estimators 中我们调整最大迭代步数(最大树的数量)。max_depth中我们调整树的深度,对于每一个树,我们设置随机树的深度。colsample_bytree, subsample中我们用一种类似bagging的采样方式, 不仅对样本进行随机采

样,也对feature进行随机采样。以上三个随机变量是在森林数量增加到某一定规模时, 残差难以拟合时, 不同训练样本和不同深度的树会有不同的效果,这样往往会带来更好的拟合。

5 评估分析与实验

5.1 评估方法

本次实验采用均方根误差（RMSE）进行评估：

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(observed_t - predicted_t)^2}$$

对参数估计值与参数真值之差平方的期望值取算术平方根。RMSE的值越小，说明结果预测的结果越好。

5.2 实验

5.2.1 参数调整

针对选择好的模型，我们对参数进行了微调，使用的方法是sklearn中的gridsearchCV [9]，过程如下：

```
tune_params = {'n_estimators':[200, 500, 1000, 2500, 5000],
               'max_depth': sp_randint(4, 10),
               'min_child_weight':[50, 100, 200, 300, 400, 500, 600],
               'colsample_bytree':sp_uniform(loc=0.8, scale=0.15),
               'subsample':sp_uniform(loc=0.8, scale=0.15),
               'eta': [0.1, 0.2, 0.3, 0.4, 0.5]}
```

图 12: 模型参数调整

最终选择出了较优的参数，即在(4.3)节中所展示的参数。

5.2.2 实验结果

模型	RMSE	训练时间
Prophet	3.81201	33h
LSTM	1.01924	1.5h
LightGBM	0.91277	1m
Xgboost	0.91163	1h

实验结果表明，Prophet的结果最差，用时也最长，说明传统的方法很难处理此类问题。LSTM可以用于处理时间序列问题，但是在这个问题上表现并不优异。以树模型为基础的XGBoost和lightGBM框架在此问题上的表现上较好且相似。虽然lightGBM的RMSE的略低，但是在用时上，lightGBM优势明显。截至到报告撰写为止，我们的成绩为351/2075，排名为16%。

6 实验改进

在这次比赛中，由于提供的数据是俄文，所以给我们在操作上带来了一定的困难。我们认为，如果对城市去做一次聚类，再按照如上的实验方法去实验的话，性能应该还能提升。因为对不不同的地域不同的城市，人民的偏好和消费水平存在显著的差异。如果要进一步改进我们的模型，那么我们要根据具体的城市，去查阅俄罗斯不同城市的地理位置以及他们在那34个月的GDP、人口等情况，新增特征去拟合。其次是我们没有定义出一些更好的除了RMSE之外的评估指标。我们尝试去定义，但效果不能达到我们的预期，于是我们舍弃了我们所定义的评估函数/

7 总结

这次我们选取的题目是playground中的一道题，之所以选择这道题是因为，时间序列的题目处理起来一向比较困难，如果选择一些简单的文本情感分类的题目，则不能很好地锻炼我们的能力。在本次竞赛中，我们前前后后尝试了四种不同的模型，从传统的方法到深度学习的方法，再到目前数据类竞赛大家都会使用的XGBoost和LightGBM。通过这次锻炼，我们对时间序列的问题如何处理有了更深的理解，对这些模型的理解也在实践中得到了提升。

参考文献

[1] S Lawrence Marple and S Lawrence Marple. *Digital spectral analysis: with applications*, volume 5. Prentice-Hall Englewood Cliffs, NJ, 1987.

[2] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

[3] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[6] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.