课　　程：2017 软件工程综合实训

项　　目：数据挖掘比赛

院　　系：数据科学与计算机学院

专　　业：软件工程

学生姓名：彭瑞

学　　号：14331226

授课教师：郑子彬，曾海标

2017 年 07 月 07 日

目录

1. 比赛介绍

本部分为比赛介绍，主要内容均来自比赛网站[1]。

1.1 比赛概况

Housing costs demand a significant investment from both consumers and developers. And when it comes to planning a budget - whether personal or corporate - the last thing anyone needs is uncertainty about one of their biggest expenses. Sberbank, Russia's oldest and largest bank, helps their customers by making predictions about realty prices so renters, developers, and lenders are more confident when they sign a lease or purchase a building.

Although the housing market is relatively stable in Russia, the country's volatile economy makes forecasting prices as a function of apartment characteristics a unique challenge. Complex interactions between housing features such as number of bedrooms and location are enough to make pricing predictions complicated. Adding an unstable economy to the mix means Sberbank and their customers need more than simple regression models in their arsenal.

In this competition, Sberbank is challenging Kagglers to develop algorithms which use a broad spectrum of features to predict realty prices. Competitors will rely on a rich dataset that includes housing data and macroeconomic patterns. An accurate forecasting model will allow Sberbank to provide more certainty to their customers in an uncertain economy.

1.2 度量方法

Submissions are evaluated on the RMSLE between their predicted prices and the actual data. The target variable, called price_doc in the training set, is the sale price of each property.

RMSLE: Root Mean Squared Logarithmic Error

$$RMSLE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\log(1+real^{(i)}) - \log(1+pred^{(i)}))^2}$$

where m is the size of the data set, $\mathrm{real}^{(i)}$ is the real price of the i-th property in the data set, and $\mathrm{pred}^{(i)}$ is the predicted price of the i-th property in the data set.

1.3 比赛奖金

1st place - $12,000
2nd place - $8,000
3rd place - $5,000

1.4 比赛日程

June 22, 2017 - Entry deadline. You must accept the competition rules before this date in order to compete.

June 22, 2017 - Team Merger deadline. This is the last day participants may join or merge teams.

June 29, 2017 - Final submission deadline.

All deadlines are at 11:59 PM UTC on the corresponding day unless otherwise noted. The competition organizers reserve the right to update the contest timeline if they deem it necessary.

1.5 比赛数据

The aim of this competition is to predict the sale price of each property. The target variable is called price_doc in train.csv.

The training data is from August 2011 to June 2015, and the test set is from July 2015 to May 2016. The dataset also includes information about overall conditions in Russia's economy and finance sector, so you can focus on generating accurate price forecasts for individual properties, without needing to second-guess what the business cycle will do.

Data Files:

train.csv, test.csv: information about individual transactions. The rows are indexed by the "id" field, which refers to individual transactions (particular properties might appear more than once, in separate transactions). These files also include supplementary information about the local area of each property.

macro.csv: data on Russia's macroeconomy and financial sector (could be joined to the train and test sets on the "timestamp" column)

sample_submission.csv: an example submission file in the correct format

data_dictionary.txt: explanations of the fields available in the other data files

BAD_ADDRESS_FIX.xlsx: Some optional extra data resolving an issue with some GIS features

1.6 比赛排行榜

To prevent overfitting, the competition's test data is partitioned into two parts, related to Public Leaderboard and Private Leaderboard respectively.

This Public Leaderboard is calculated with approximately 35% of the test data, while the Private Leaderboard is calculated with the other 65% of the test data. The Private Leaderboard reflects the final standings, and may be different with the Public Leaderboard.

2. 比赛环境配置

2.1 主要运行环境

Python - a programming language that lets you work quickly and integrate systems more effectively.[4]

XGBoost - an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.[5]

Jupyter Notebook - an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.[6]

2.2 运行环境配置总结

在 Windows 操作系统下使用 Anaconda 作为 Python 的运行环境，Anaconda 已经预装有大量可以用于数据挖掘的库，同时也预装了 Jupyter Notebook，上手非常方便，但是安装 XGBoost 的时候遇到了不小的麻烦，安装过程中参考了大量的资料，其中主要参考的是这个网站提供的资料：

https://www.ibm.com/developerworks/community/blogs/jfp/entry/Installing_XGBoost_For_Anaconda_on_Windows?lang=en

但是即使能够在 Windows 下运行 XGBoost，依然与 Kernel 上面的结果有不小的差距，后来安装 VMWare 下的 Ubuntu 17.04 虚拟机，使用虚拟机运行代码，有明显的改善。

2.3 试运行 XGBoost 程序

```
# import
import pandas as pd
import xgboost as xgb
import sklearn as skl

# load data
train = pd.read_csv('../input/train.csv')
test = pd.read_csv('../input/test.csv')

# encode data
def encode(df):
    df = df.drop(['id', 'timestamp'], 1)
    for c in df.columns:
        if df[c].dtype == 'object':
            df[c] = skl.preprocessing.LabelEncoder().fit_transform(list(df[c].values))
    return df
dtrain = xgb.DMatrix(encode(train.drop(['price_doc'], 1)), train['price_doc'])
dtest = xgb.DMatrix(encode(test))

# train and get result
param = {'eta': .05, 'max_depth': 5, 'objective': 'reg:linear', 'eval_metric': 'rmse'}
result = pd.DataFrame({'id': test.id, 'price_doc': xgb.train(param, dtrain, 500).predict(dtest)})
result.to_csv('xgb.csv', index=0)
```

使用简化的 XGBoost 代码[2]运行并提交，结果约为 0.316，这一模型代码简洁，忽略宏观经济与时间戳，直接使用 LabelEncoder 进行非数字信息的处理，使用的参数也非常随意，未进行交叉验证。

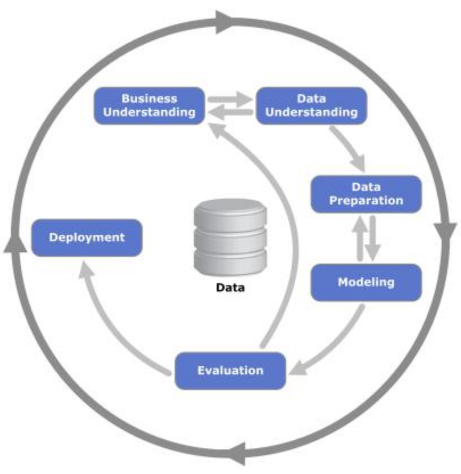
这样的结果至少能够说明 XGBoost 的环境配置成功了，虽然还有很大的提升空间。

3. 解决方案

3.1 数据挖掘解决方案的基本流程

Cross Industry Standard Process for Data Mining

- Business Understanding
  - To learn how house prices are determined and predict house prices
- Data Understanding
  - A Very Extensive Sberbank Exploratory Analysis
- Data Preparation
  - Missing values imputation
  - Feature engineering
- Modeling
  - XGB, LightGBM, Random Forest
- Evaluation
  - Local CV
  - Feature importance
- Deployment



Introduction and image source: [3]

3.2 数据理解

我们已经理解了本次比赛的任务为预测房价，接下来便是对数据的理解。Extensive Sberbank Exploratory Analysis 提供了对数据的一些分析结果[2]，在这里用作参考。

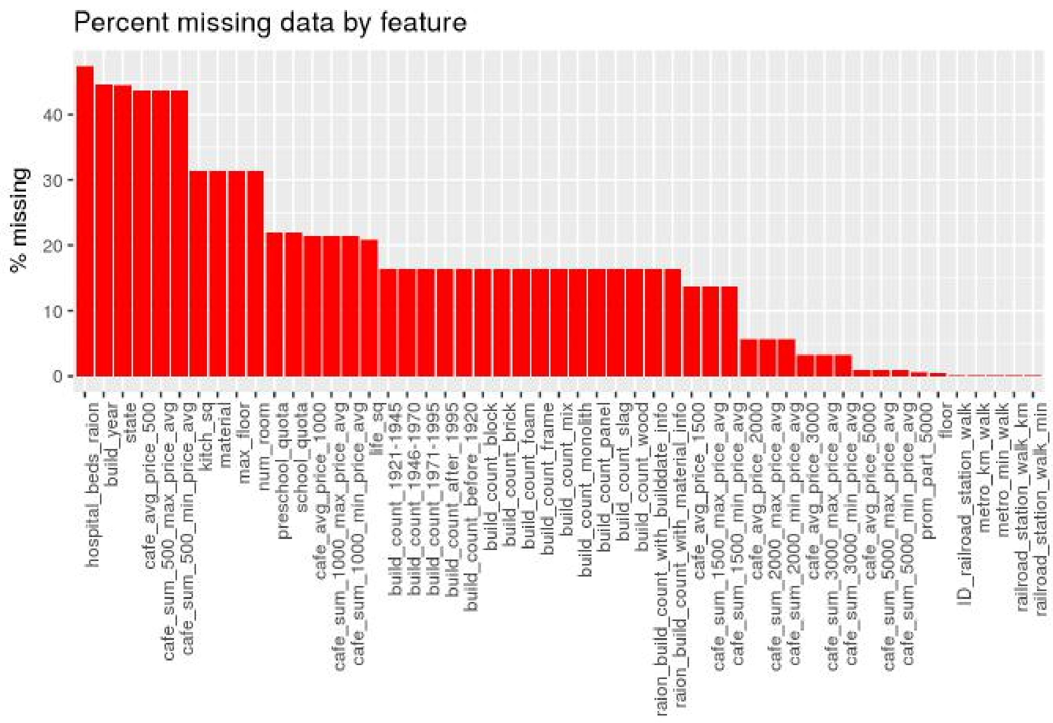


Image source: [2]

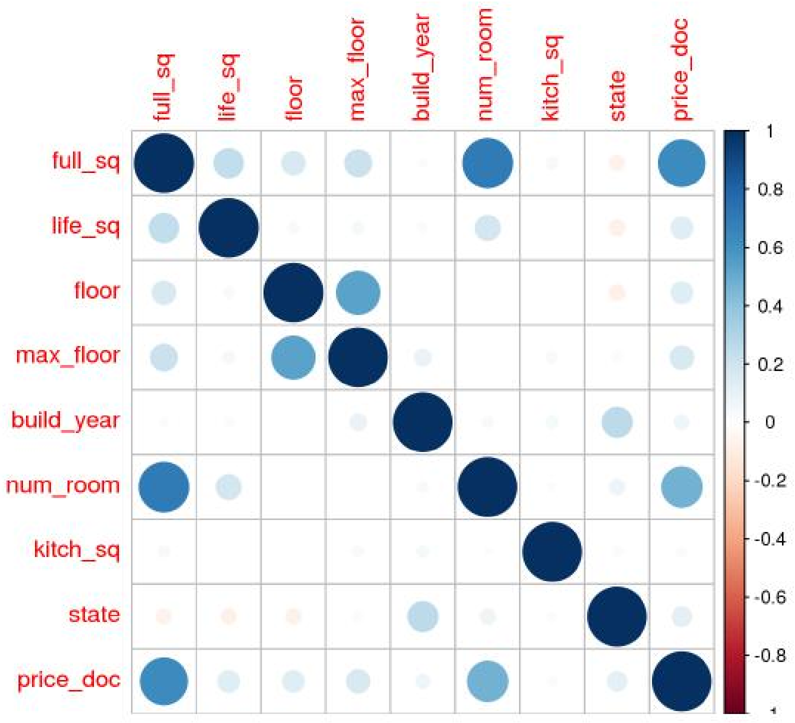
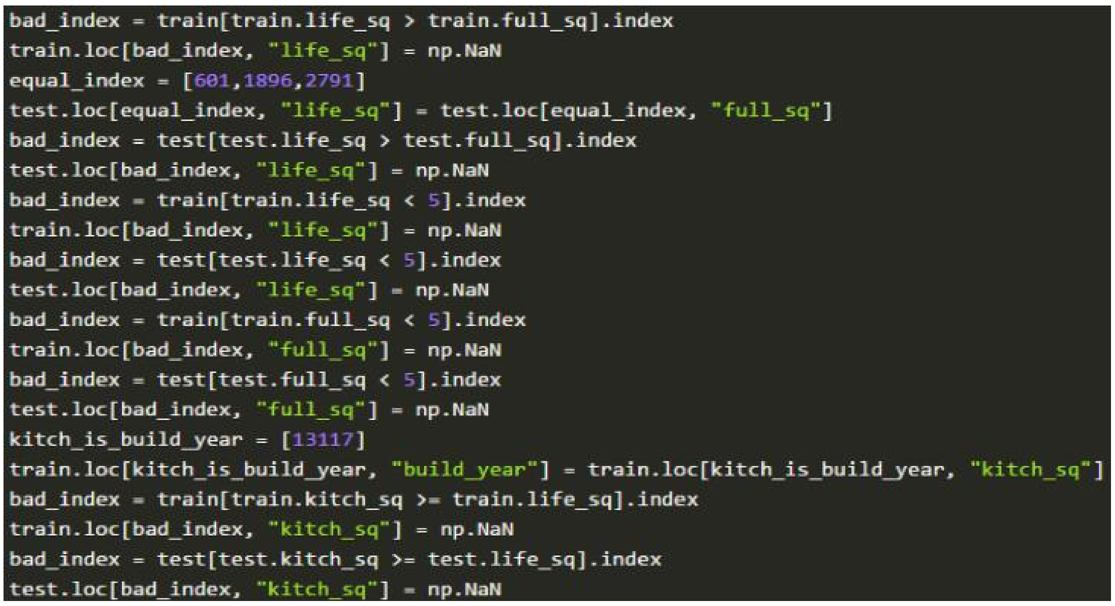


Image source: [2]

3.3 数据准备

对数据的分析为数据预处理提供了指导，在 Kernel 上已有成型的数据预处理方法，节选如下：

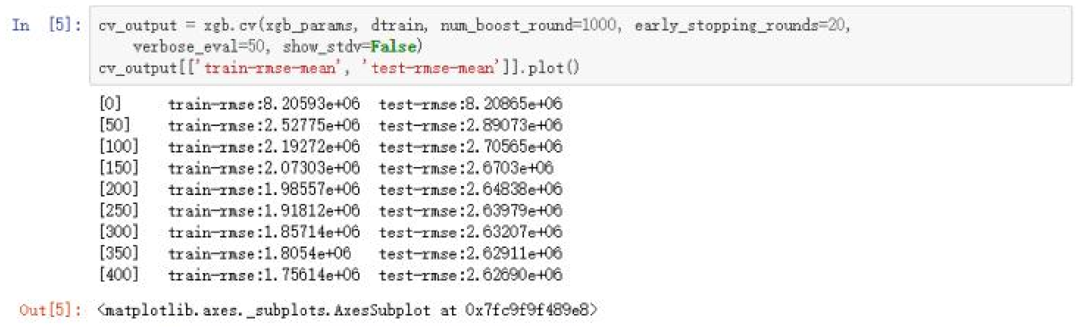


```
bad_index = train[train.life_sq > train.full_sq].index
train.loc[bad_index, "life_sq"] = np.NaN
equal_index = [601,1896,2791]
test.loc[equal_index, "life_sq"] = test.loc[equal_index, "full_sq"]
bad_index = test[test.life_sq > test.full_sq].index
test.loc[bad_index, "life_sq"] = np.NaN
bad_index = train[train.life_sq < 5].index
train.loc[bad_index, "life_sq"] = np.NaN
bad_index = test[test.life_sq < 5].index
test.loc[bad_index, "life_sq"] = np.NaN
bad_index = train[train.full_sq < 5].index
train.loc[bad_index, "full_sq"] = np.NaN
bad_index = test[test.full_sq < 5].index
test.loc[bad_index, "full_sq"] = np.NaN
kitch_is_build_year = [13117]
train.loc[kitch_is_build_year, "build_year"] = train.loc[kitch_is_build_year, "kitch_sq"]
bad_index = train[train.kitch_sq >= train.life_sq].index
train.loc[bad_index, "kitch_sq"] = np.NaN
bad_index = test[test.kitch_sq >= test.life_sq].index
test.loc[bad_index, "kitch_sq"] = np.NaN
```

Image source: [2]

3.4 建模

本次比赛使用的基本模型以 XGBoost 为主，当然也可以使用 LightGBM 以及随机森林等模型进行补充，对每个模型都可以输出结果，以用于接下来的效果评估与模型融合。以下为 XGBoost 的训练与检验过程参考输出。



```
In  [5]: cv_output = xgb.cv(xgb_params, dtrain, num_boost_round=1000, early_stopping_rounds=20,
             verbose_eval=50, show_stdv=False)
         cv_output[['train-rmse-mean', 'test-rmse-mean']].plot()

         [0]     train-rmse:8.20593e+06  test-rmse:8.20865e+06
         [50]    train-rmse:2.52775e+06  test-rmse:2.89073e+06
         [100]   train-rmse:2.19272e+06  test-rmse:2.70565e+06
         [150]   train-rmse:2.07303e+06  test-rmse:2.6703e+06
         [200]   train-rmse:1.98557e+06  test-rmse:2.64838e+06
         [250]   train-rmse:1.91812e+06  test-rmse:2.63979e+06
         [300]   train-rmse:1.85714e+06  test-rmse:2.63207e+06
         [350]   train-rmse:1.8054e+06   test-rmse:2.62911e+06
         [400]   train-rmse:1.75614e+06  test-rmse:2.62690e+06

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc9f9f489e8>
```

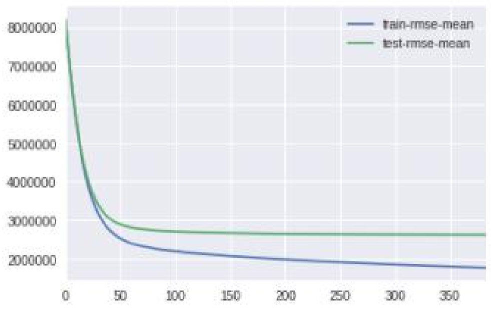
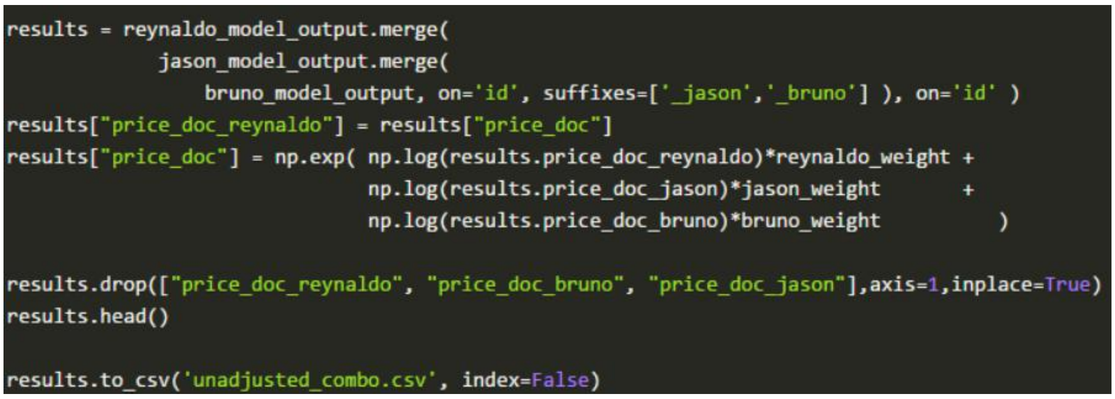


Image source: [2]

3.5 融合模型

对不同的（最好相差较大）优秀模型进行融合，可能会起到更好的效果，以下为融合模型的参考代码。



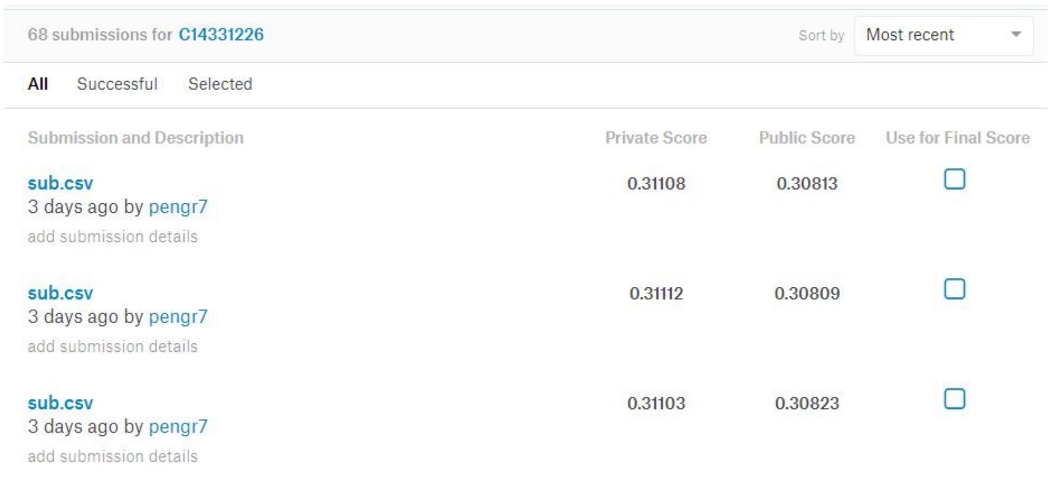
```
results = reynaldo_model_output.merge(
            jason_model_output.merge(
                bruno_model_output, on='id', suffixes=['_jason','_bruno'] ), on='id' )
results["price_doc_reynaldo"] = results["price_doc"]
results["price_doc"] = np.exp( np.log(results.price_doc_reynaldo)*reynaldo_weight +
                               np.log(results.price_doc_jason)*jason_weight       +
                               np.log(results.price_doc_bruno)*bruno_weight          )

results.drop(["price_doc_reynaldo", "price_doc_bruno", "price_doc_jason"],axis=1,inplace=True)
results.head()

results.to_csv('unadjusted_combo.csv', index=False)
```

Image source: [2]

3.6 提交结果



68 submissions for C14331226 Sort by Most recent

All Successful Selected

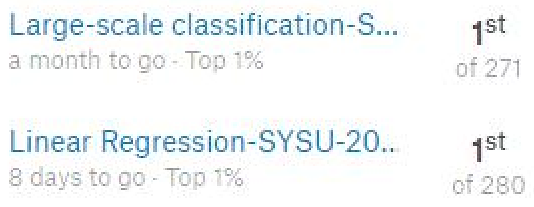
Submission and Description	Private Score	Public Score	Use for Final Score
sub.csv 3 days ago by pengr7 add submission details	0.31108	0.30813	☐
sub.csv 3 days ago by pengr7 add submission details	0.31112	0.30809	☐
sub.csv 3 days ago by pengr7 add submission details	0.31103	0.30823	☐

4. 实训总结

关于此次实训的一些技术上方法和技巧，我已在用于展示的文档里给出。在这篇实训总结中，我想谈谈我在数据挖掘上的一些摸索。

在寒假我参加了数学建模的美赛，这学期我有一门数据挖掘的课程，除此以外，我在 Coursera 上完成了 Machine Learning 课程的学习并满分通过。在对有关理论有了一些了解和思考后，我开始了数据挖掘的实战。数据挖掘课程的课内竞赛(Kaggle in Class)是非常好的锻炼机会，经过大量的摸索和尝试，我在两次课内竞赛中均取得了第一名，并积累了大量的数据挖掘实战经验，包括尝试线性模型、神经网络、支持向量机、集成学习、XGBoost 等多种方法。尤其是 XGBoost 的调

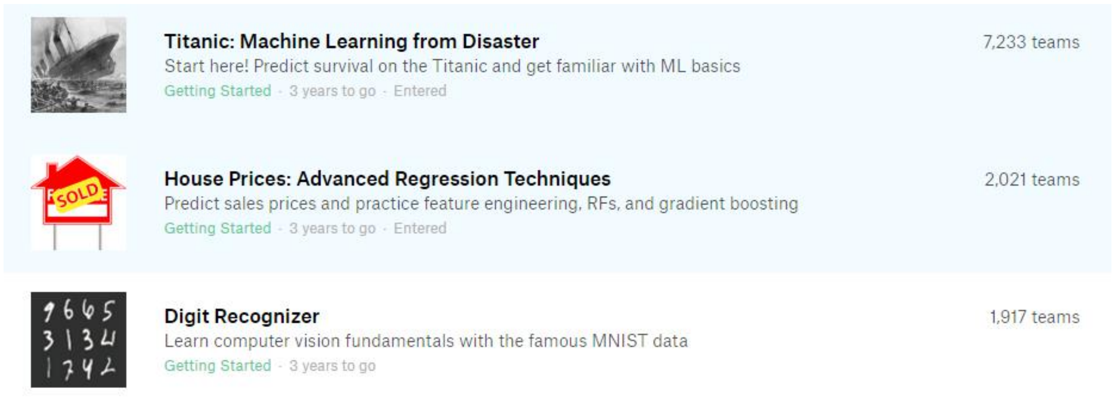
参，对于不同的数据集，最适宜的参数也可能不同。把若干个结果以适当的方式融合起来是使自己成绩脱颖而出的关键。每天的提交次数限制也十分让人恼火，因此最好要在本地跑出不错的结果，对自己结果有足够信心之后，再去尝试提交。但有时候可能也需要去测试一种方法或模型在榜单上的分数如何，使用一次提交机会也在所难免。



Large-scale classification-S...
a month to go · Top 1%
1st
of 271

Linear Regression-SYSU-20...
8 days to go · Top 1%
1st
of 280

对我而言，这两次课内竞赛毕竟只是进入数据挖掘领域的敲门砖，真实世界的数据很可能比这些经过预处理的数据复杂，因此我开始转向 Kaggle 上的数据挖掘竞赛，从入门级别的竞赛开始做起，仔细研究这些竞赛的 Tutorial 部分的每一个 Notebook 的技巧和原理，在这个过程中，我熟悉了 Python 的一些使用方法，也更加了解了数据挖掘竞赛的“解题思路”。



Titanic: Machine Learning from Disaster
Start here! Predict survival on the Titanic and get familiar with ML basics
Getting Started · 3 years to go · Entered
7,233 teams

House Prices: Advanced Regression Techniques
Predict sales prices and practice feature engineering, RFs, and gradient boosting
Getting Started · 3 years to go · Entered
2,021 teams

Digit Recognizer
Learn computer vision fundamentals with the famous MNIST data
Getting Started · 3 years to go
1,917 teams

这些学习的过程和这次预测房价的比赛是穿插着进行的。在这次比赛中，我使用了一个非常简单的 XGB 模型作为尝试，交上去的结果已经能够达到 0.316 左右了。特征工程和模型融合是非常重要的环节，因为倘若有了现成的数据，很多事都可以借助 XGBoost 来完成，但是特征工程和模型融合不行。Kernels 和 Discussion 部分的代码和讨论给了我非常大的启发。经过不断地学习与改良，截至 7 月 7 日我的最好结果已经达到了 0.311(private)/0.308(public)左右。

在数据挖掘实践的过程中，我遇到了很多问题，有的问题很快得到了解决，有的问题困扰我的时间较长，在这里我列出数据挖掘竞赛中遇到的主要问题与体会：

1. 我本来倾向于用 MATLAB 进行数据处理，而对 Python 的了解十分有限，但是出于实用、内存等原因，我不得不从实践中逐渐学习 Python 的使用。
2. XGBoost 的安装遇到了很多麻烦，通过查看官方文档并没有有效解决问题，最终我查阅了大量资料，完成了安装。其中 MINGW-W64 的版本问题，使用 pip 安装的问题令我印象深刻。Anaconda 是 Windows 下 Python 开发的不错环境。
3. 即使是同样的代码，在本地运行和 Kernel 运行的结果仍有较大差异，这可能是由于本地与 Kernel 在环境配置上的差异，而 XGBoost 对某些参数差异“异常敏感”。
4. XGBoost 的参数、融合的参数，看起来相当“武断”，因为很多是根据提

交后的结果来调整的，这可能造成在 Public Leaderboard 上的过拟合，但是根据往常的经验来看，当 Private Leaderboard 公布后，成绩也许会“变糟”，但是排名一般不会有较大变化。

我希望自己在数据挖掘领域的浓厚兴趣能够转化为对数据的“洞察力”和能够给出切实有效的数据科学解决方案的能力。我希望今后能够继续深化在数据科学领域的学习与研究，这是我理想的未来研究方向。

5. 附录

参考资料

[1] https://www.kaggle.com/c/sberbank-russian-housing-market

[2] https://www.kaggle.com/c/sberbank-russian-housing-market/kernels

[3]https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining

[4] https://www.python.org/

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

[6] http://jupyter.org/

[7]https://www.ibm.com/developerworks/community/blogs/jfp/entry/Installing_XGBoost_For_Anaconda_on_Windows?lang=en

GitHub 目录

https://github.com/pengr7/Sberbank-Kaggle