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：　周小峰

学　　号：　14331393

授课教师：　郑子彬，曾海标

2017 年 7 月 2 日

目录

摘要

本次数据挖掘比赛是Kaggle的平台上的Sberbank Russian Housing Market，目的是预测Russia的房价，是典型的回归问题。本次实验报告将从赛事简介、评估方法、数据可视化、数据清洗、特征选择及特征工程、模型选择和最终成绩这几个方面来介绍这次实验。

关键词：housing predict， xgboost， 特征工程，回归

1. 比赛简介

Sberbank Russian Housing Market 这个比赛是让我们在 Russia 的不稳定经济条件下预测房价的波动。比赛所包含的数据除了常规的训练集、测试集，还给出了一个 macro 文件，主要是 Russia 宏观经济和财务信息。

2. 评估方法

一般涉及到回归类问题的比赛，所使用的评估方式都是 RMSE，不过由于这次比赛所预测的房价的范围较大，RMSE 容易被一些大的值误导，从而导致如果一个非常大的值预测不准确，RMSE 会很大，所以这次比赛使用的评估方法是 RMSLE（Root Mean Squared Logarithmic Error），它将结果先取 log 然后在求 RMSE，能够稍微解决数据分布不均的影响。

$$\varepsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

其中 ε 是 RMSLE 的结果也就是误差

n 是数据集的数量

p_i 是预测值

a_i 是真实值

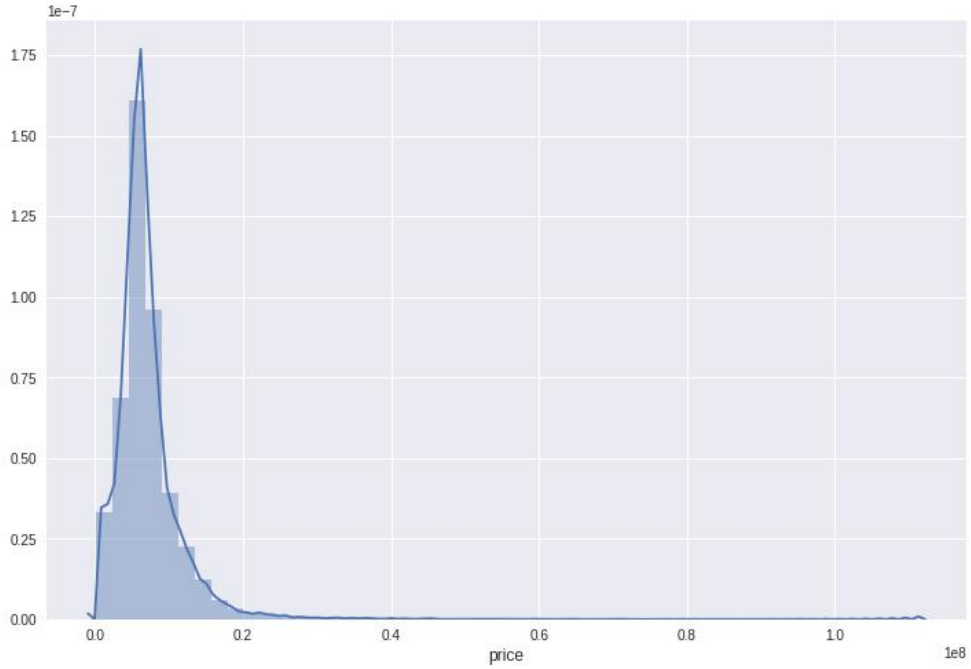
log(x)就是 x 的自然对数

3. 数据可视化

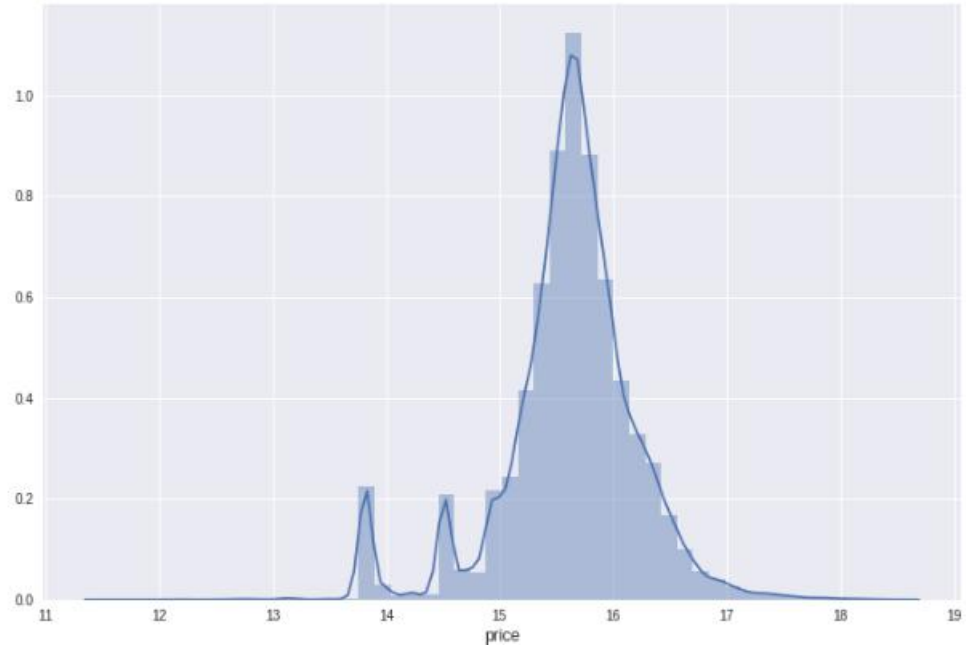
3.1 房价信息

```
train_df.price_doc.describe()
```

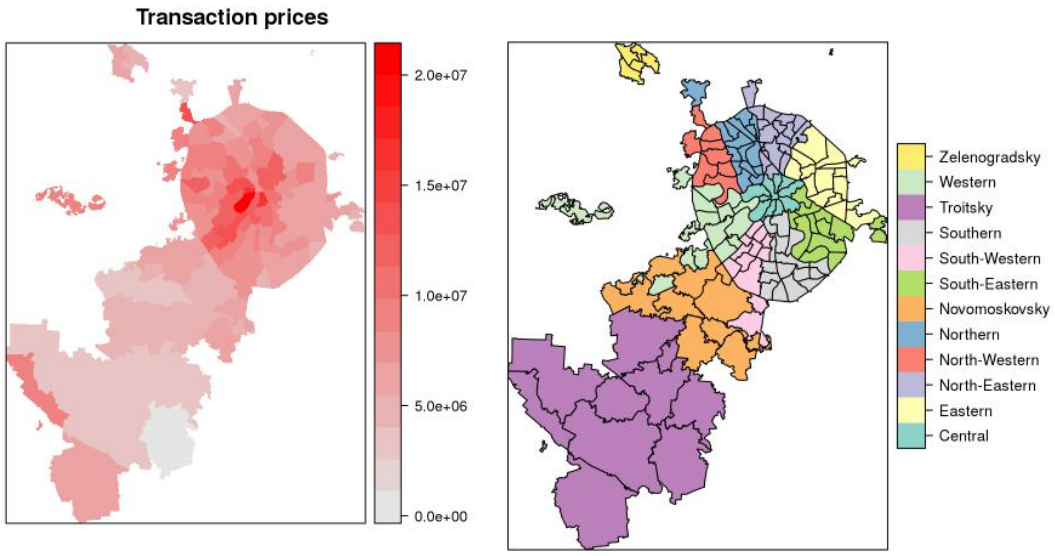
```
count    3.047100e+04
mean     7.123035e+06
std      4.780111e+06
min      1.000000e+05
25%      4.740002e+06
50%      6.274411e+06
75%      8.300000e+06
max      1.111111e+08
Name: price_doc, dtype: float64
```



从房价的分布来看，是很不均匀的、有峰值的、偏正态化的，对房价进行 log 转换，使其分布较为正态化。

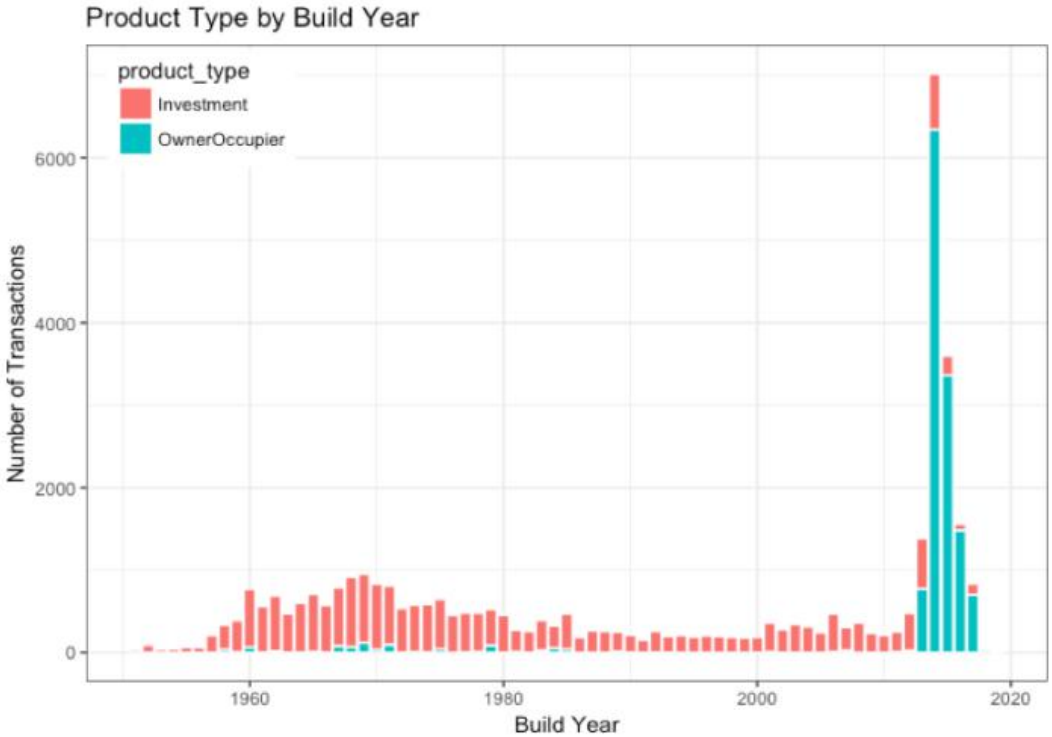


3.2 交易次数和地理位置



上面左边的图显示了各地区交易价钱，颜色的深浅代表着价格的高低，右边的图根据 sub_area 将整个区域划分为 12 个区，通过左右两张图的结合，可以发现，在地图右上角的类圆形区域中，平均房价是较高的。而在左下角部分区域，通过网上查阅资料得知是 Russia 的新区，因此地理位置可能是一个影响价格的重要因素。

3.3 product_type



Product_type 主要有两种类型，一种是 Investment，另一种就是 OwnerOccupier，在这次比赛当中，有人提出了一个 magic number，是基于 Investment 的，训练模型的时候，根据 Investment 和 OwnerOccurpier 两种类型进行分开训练预测，最终合并，在 public LB 上分数虽然会下降，但是在 Private LB 上的分数却有不少提高。

3.4 缺失值统计

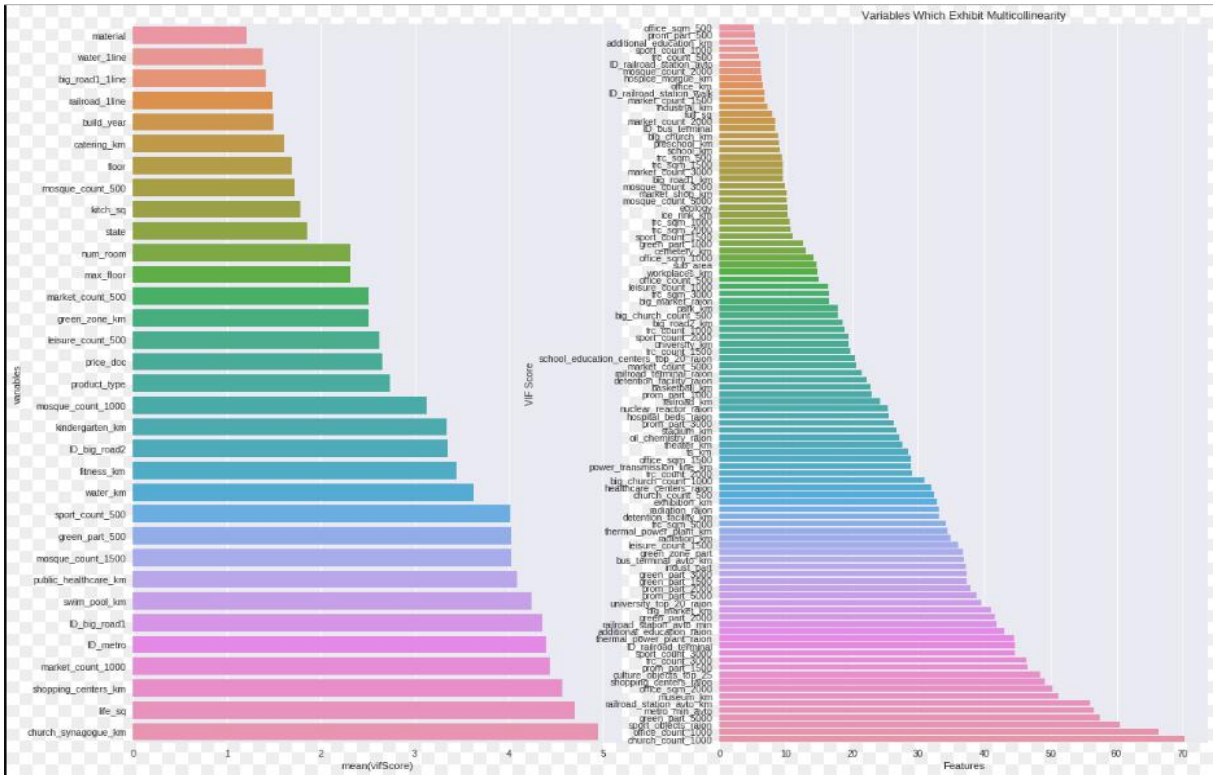
训练集和测试集缺失数据统计如下：

	Total	Percent
hospital_beds_raion	14441	0.473926
build_year	13605	0.446490
state	13559	0.444980
cafe_avg_price_500	13281	0.435857
cafe_sum_500_max_price_avg	13281	0.435857
cafe_sum_500_min_price_avg	13281	0.435857
max_floor	9572	0.314135
material	9572	0.314135
num_room	9572	0.314135
kitch_sq	9572	0.314135
preschool_quota	6688	0.219487
school_quota	6685	0.219389
cafe_sum_1000_min_price_avg	6524	0.214105
cafe_sum_1000_max_price_avg	6524	0.214105
cafe_avg_price_1000	6524	0.214105
life_sq	6383	0.209478
build_count_frame	4991	0.163795
build_count_1971-1995	4991	0.163795
build_count_block	4991	0.163795
raion_build_count_with_material_info	4991	0.163795

	Total	Percent
hospital_beds_raion	3418	0.446098
cafe_sum_500_min_price_avg	3159	0.412294
cafe_avg_price_500	3159	0.412294
cafe_sum_500_max_price_avg	3159	0.412294
preschool_quota	1596	0.208301
school_quota	1595	0.208170
cafe_sum_1000_min_price_avg	1222	0.159488
cafe_sum_1000_max_price_avg	1222	0.159488
cafe_avg_price_1000	1222	0.159488
build_count_1946-1970	1218	0.158966
build_count_before_1920	1218	0.158966
build_count_1921-1945	1218	0.158966
build_count_mix	1218	0.158966
build_count_1971-1995	1218	0.158966
build_count_block	1218	0.158966
raion_build_count_with_builddate_info	1218	0.158966
raion_build_count_with_material_info	1218	0.158966
build_count_after_1995	1218	0.158966
build_count_wood	1218	0.158966

通过观察两张缺失数据统计表，发现在训练集和测试集中相同特征的缺失率相差不大，数据之所以缺失，很大原因可能是因为这些特征是买房时人们不太关心的，那么，所以可以删除一些缺失率过高并且对房价没什么影响的数据，降低数据的维度。

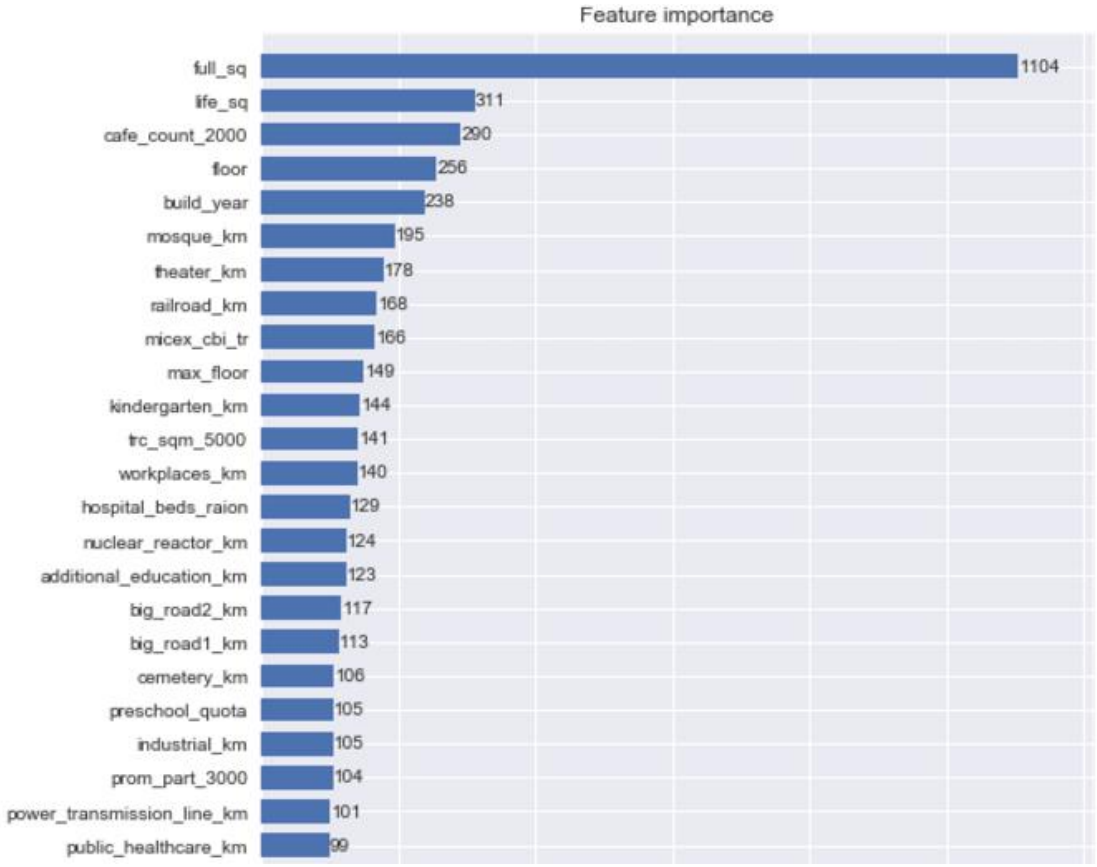
3.5 特征共线性



本次训练集有 290 多个特征，经过分析，得出一张特征共线性程度的表，对于那些共线性程度较严重的特征，在进行数据清洗的时候，可以删除一些，从而降低数据维度，减少过拟合。

3.6 特征重要性

在不经过任何数据清洗和没有做任何特征工程的前提下，利用 xgboost 对训练集进行训练，得到如下特征重要性图：



从图中看出，有关房子信息比如：full_sq、life_sq、floor、build_year 等对房子价格有着很大影响。

网上 Kernel 中有很多人都对数据进行了可视化如 SRK、TroyWalters、JTremoureux 等，并且都很详细。

4. 数据清洗

这个比赛中的数据不仅缺失率高，而且有一些数据是明显不合理的，比如 life_sq（生活面积）比 full_sq（房屋总面积）大，有的建造的年份为 1,2 或者 4095 等，因此对于那些不合理的数据进行清洗是很有必要的。在 Kernel 上有人给出了一份较好的数据清洗脚本，于是我在那份脚本的基础上，还进行了一些必要的处理。

4.1 根据单位价格删除一些极端的数据

```
train = train[train.price_doc/train.full_sq <= 600000]
train = train[train.price_doc/train.full_sq >= 10000]
```

根据每平方米的价格，将那些小于 10000 大于 600000 的数据删除.

4.2 根据 full_sq、life_sq 和 kitch_sq 之间的关系，利用均值填充 life_sq 和 kitch_sq 中的缺失值。

```
full_life_ratio = (train['full_sq'] / train['life_sq']).mean()
train['life_sq'] = train['life_sq'].fillna(train['full_sq'] / full_life_ratio)
full_life_ratio = (test['full_sq'] / test['life_sq']).mean()
test['life_sq'] = test['life_sq'].fillna(test['full_sq'] / full_life_ratio)

full_kitchen_ratio = (train['full_sq'] / train['kitch_sq']).mean()
train['kitch_sq'] = train['kitch_sq'].fillna(train['full_sq'] / full_kitchen_ratio)
full_kitchen_ratio = (test['full_sq'] / test['kitch_sq']).mean()
test['kitch_sq'] = test['kitch_sq'].fillna(test['full_sq'] / full_kitchen_ratio)
```

4.3 根据缺失率和共线性程度删除一些意义的特征

```
train = train.drop(['culture_objects_top_25_raion', 'oil_chemistry_raion', 'railroad_terminal_raion', '
    nuclear_reactor_raion', 'build_count_foam', 'big_road1_1line', 'railroad_1line', 'office_sqm_500',
    'trc_sqm_500', 'cafe_count_500_price_4000', 'cafe_count_500_price_high', 'mosque_count_500', '
    leisure_count_500', 'office_sqm_1000', 'trc_sqm_1000', 'cafe_count_1000_price_high', '
    mosque_count_1000', 'cafe_count_1500_price_high', 'mosque_count_1500', 'cafe_count_2000_price_high'
    ], axis=1)
test = test.drop(['culture_objects_top_25_raion', 'oil_chemistry_raion', 'railroad_terminal_raion', '
    nuclear_reactor_raion', 'build_count_foam', 'big_road1_1line', 'railroad_1line', 'office_sqm_500',
    'trc_sqm_500', 'cafe_count_500_price_4000', 'cafe_count_500_price_high', 'mosque_count_500', '
    leisure_count_500', 'office_sqm_1000', 'trc_sqm_1000', 'cafe_count_1000_price_high', '
    mosque_count_1000', 'cafe_count_1500_price_high', 'mosque_count_1500', 'cafe_count_2000_price_high'
    ], axis=1)
```

5. 特征选择及特征工程

这次比赛所做的特征工程很少，主要是对时间戳 timestamp 和房子本身一些相关特征进行了一些处理。

5.1 Timestamp

从 timestamp 中提取出一些信息如 yearmonth、month、day_of_week 以及根据 yearmonth 进行统计。

```
# Add month-year
month_year = (train.timestamp.dt.month + train.timestamp.dt.year * 100)
month_year_cnt_map = month_year.value_counts().to_dict()
train['month_year_cnt'] = month_year.map(month_year_cnt_map)

month_year = (test.timestamp.dt.month + test.timestamp.dt.year * 100)
month_year_cnt_map = month_year.value_counts().to_dict()
test['month_year_cnt'] = month_year.map(month_year_cnt_map)

# Add week-year count
week_year = (train.timestamp.dt.weekofyear + train.timestamp.dt.year * 100)
week_year_cnt_map = week_year.value_counts().to_dict()
train['week_year_cnt'] = week_year.map(week_year_cnt_map)

week_year = (test.timestamp.dt.weekofyear + test.timestamp.dt.year * 100)
week_year_cnt_map = week_year.value_counts().to_dict()
test['week_year_cnt'] = week_year.map(week_year_cnt_map)

# Add month and day-of-week
train['month'] = train.timestamp.dt.month
train['dow'] = train.timestamp.dt.dayofweek

test['month'] = test.timestamp.dt.month
test['dow'] = test.timestamp.dt.dayofweek
```

5.2 房子本身特征

根据 floor 和 max_floor 计算房子相对高度，根据 kitch_sq 和 full_sq 计算厨房相对大小等。

```
# Other feature engineering
train['rel_floor'] = train['floor'] / train['max_floor'].astype(float)
train['rel_kitch_sq'] = train['kitch_sq'] / train['full_sq'].astype(float)

test['rel_floor'] = test['floor'] / test['max_floor'].astype(float)
test['rel_kitch_sq'] = test['kitch_sq'] / test['full_sq'].astype(float)

train.apartment_name=train.sub_area + train['metro_km_avto'].astype(str)
test.apartment_name=test.sub_area + train['metro_km_avto'].astype(str)

# modify 6.27
train['room_size'] = train['life_sq'] / train['num_room'].astype(float)
test['room_size'] = test['life_sq'] / test['num_room'].astype(float)
```

5.3 macro 宏观信息

Macro 文件当中给出了 Russia 的经济相关信息，但是我只取了其中特征 VIF（方差膨胀因子）小于 6 的一些特征，至于如何计算特征 VIF，可以参考 Cro-Magno 的 Kernel。

```
##  [1] "balance_trade"
##  [2] "balance_trade_growth"
##  [3] "eurrub"
##  [4] "average_provision_of_build_contract"
##  [5] "micex_rgbi_tr"
##  [6] "micex_cbi_tr"
##  [7] "deposits_rate"
##  [8] "mortgage_value"
##  [9] "mortgage_rate"
## [10] "income_per_cap"
## [11] "rent_price_4.room_bus"
## [12] "museum_visitis_per_100_cap"
## [13] "apartment_build"
```

5.4 Latitude 和 longitude

对于经纬度，官方给出的数据中是没有具体给出的，只是通过 sub_area 描述了大致位置，Kernel 上的 Chippy 对位置的经纬度做了精细的处理，并将处理后的结果公开，我便直接将其加入到我的训练和测试数据集中。

5.5 magic number

Magic number 是 Kernel 上 Andy Harless 根据 Russia 近几年每个季度房价波动变化得出来的，下面两张图显示了 Russia 近几年价格变化，我们所要预测的是 2015-2016 年房价，此时的房价实际上是偏低的，所以将训练集的 price_doc*0.969，在进行训练所得到的结果误差会降低不少。

House price change

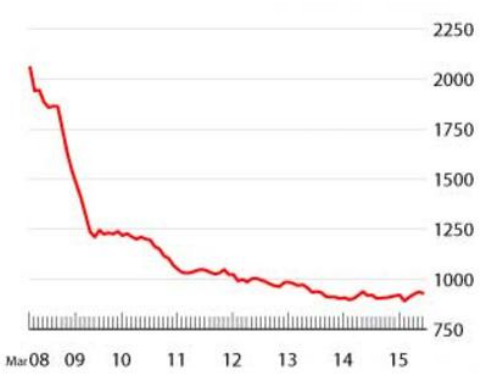
% change over a year earlier

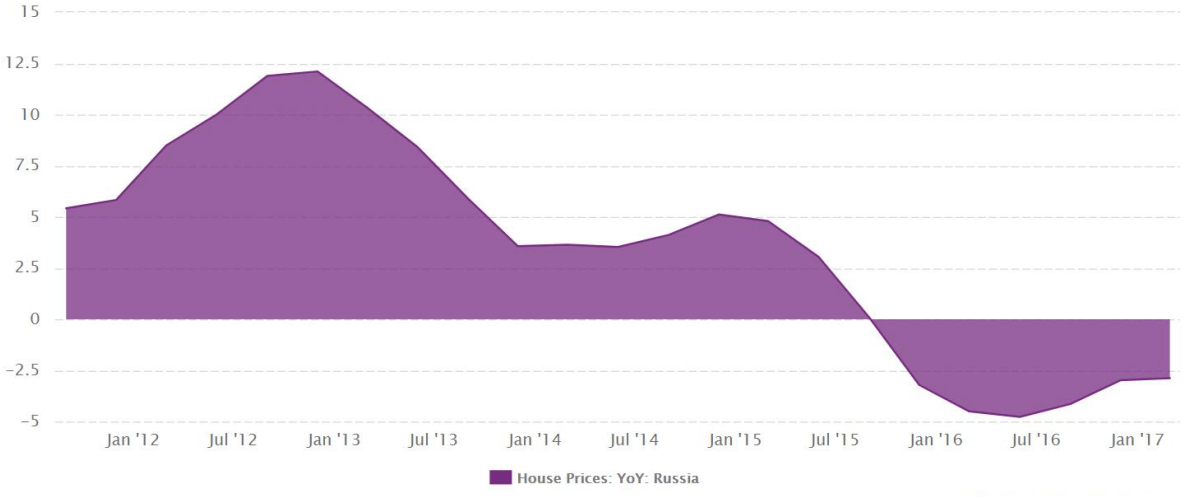
	Q1	Q2	Q3	Q4
2016	6.52	0.69		
2015	5.05	-0.32	3.32	-1.35
2014	4.45	-3.62	0.88	-2.30
2013	-2.04	-2.51	-2.46	-1.43
2012	1.21	-1.60	-2.43	1.56
2011	0.97	-0.86	1.35	-5.44
2010	-1.38	-3.96	-5.07	-6.43
2009	-14.48	-5.90	-1.53	0.16
2008		-8.36	-1.22	-16.96

% change over a quarter (QoQ)

Source: imobiliare.ro

Average Selling Price of Apartments





6. 模型选择

这次比赛中我所选择的模型只有 xgboost 一个，最终的结果是由四个 xgboost 得到的结果按照一定的权重结合得出的。

第一个 xgboost 模型没有使用 macro 文件中的宏观信息，只是做了一些简单的清洗和特征工程，所得到的结果在 public 上是 0.31794，private 上是 0.32355.

第二个 xgboost 模型加入了 macro 文件中的宏观信息，但是没有做任何数据清洗和特征工程，所得到的结果在 public 上是 0.31655，Private 上是 0.32265.

第三个 xgboost 模型没有使用 macro 文件中的宏观信息，在 price_doc 上用了 Andy Harless 所提到的 magic number，以及对年分和季度对 price_doc 进行

了一些相关处理。所得到的结果在 public 上是 0.31324， 在 Private 上是 3.31728.

前面三个模型的结果我按照 xgb1 * 0.2 + xgb2 * 0.2 + xgb3 * 0.6 的权重进行平均结合，在 public 上是 0.31045， 在 Private 上是 0.31657.

第四个 xgboost 模型没有使用 macro 文件中的宏观信息，只是在第一个 xgboost 的基础上添加了同学们课堂分享时所提到的一些特征以及一些数据清洗方法，单模型结果在 public 上是 0.31286， Private 上是 0.31770.

我将前面三个模型按权重得到的结果和第四个模型得到的结果按照 0.65，0.35 的方式在进行加权平均，得到最终结果在 public 上是 0.30975， Private 上是 0.31484. 这个结果最终的排名是 80/3462.

7. 总结

在比赛结束后，看了排名第二的选手 SchoolPal 所写的 Kernel，对于数据清洗，他清除了很多没有必要的特征，这部分比我做的多得多，但是对于特征工程，他却没怎么做，只是简单的从 timestamp 中提取 yearmonth、month，对 floor、kitchen_sq、build_age 进行简单处理，这些还不如我做的多。他在模型选择方面倒是用了多个模型，xgboost、lightGBM、DNN 等以及最终的 stacking. 和他简单比较后，我觉得我所欠缺的是尝试，我应该去尝试多个不同模型，虽然 xgboost 本身不错，但只用一个所得到的结果还是免不了局限性。

从 5 月 1 号正式开始加入这个比赛到 6 月 29 号结束，在将近 2 个月的时间当中，通过老师知识讲解和师兄们比赛经验分享、同学们展示，我学到了很多东西，在做比赛的过程中，虽然是个人参赛，但是 Kernel 和 discuss 这两个区中有着大量的经验和知识分享，只要善于利用，一定会有进步。

参考文献

1. https://www.kaggle.com/aharless/latest-iteration-in-this-silly-game。
2. https://www.kaggle.com/schoolpal/nn-stacking-magic-no-magic-30409-private-31063.
3. https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-sberbank.
4. https://www.kaggle.com/captcalculator/a-very-extensive-sberbank-exploratory-analysis.
5. https://www.kaggle.com/arathee2/creating-some-useful-additional-features.
6. https://www.kaggle.com/nigelcarpenter/property-location-attempt-3.
7. https://www.kaggle.com/robertoruiz/dealing-with-multicollinearity.

比赛代码：https://github.com/xfsysu/Sberbank-Russian-Housing-Market.