2018 春季学期金融数据分析与数据挖掘

期末项目

项目名称：房价预测

小组成员：

11611512 王丹妤

11510378 李莉欣

11510590 高宇馨

一．概况

1. 问题概述

预测住宅价格可以指导房地产经纪人在决策和定价过程中，帮助客户为自己的房屋找到最优的销售价格。行业领先者也将房价预测用于各种目的。Zillow 利用房价预测来指导网上卖家和购买者。Airbnb 利用房价预测来通知主人更好地定价他们的地方。

该项目的目的是预测房屋的最终价格，项目给出了 79 个解释变量，用来描述爱荷华州艾姆斯住宅的每个方面。这个项目介绍及数据均来自于 Kaggle 中的比赛。（https://www.kaggle.com/c/house-prices-advanced-regression-techniques#description）

2. 问题解析

这个项目的最终目标是根据给定的数据确定一个模型，以使得预测的房价尽可能准确。这个问题的相关任务包括：

（1）已有的数据分析-了解数据集中的特征类型、不同特征的缺失值、存在多少异常值以及特定特征是否倾斜，这第一步对于最终创建一个良好的模型至关重要。

（2）数据预处理-利用从已有数据分析中获得的信息对数据进行预处理，可能包括特征转换、数据类型转换、异常值检测和估算缺失值。

（3）基准建模-利用已有的基础模型建模，以便为将来的建模改进建立基准。

（4）模型改进-将上述基准模型以不同组合形式合并，调整模型参数以提高单个模型和组合模型的性能，得到最优结果。

二．数据探索

首先打开 train.csv 文件，发现共 78 个参量可能影响房价，遵循从整体到局部的分析方法来对数据进行探索和可视化处理。

在已给出的“training”数据集中，包含分类特征和数值特征，以及一些缺失的值。具体而言，共有 1460 个观测值，每一个都有 79 个特征，包括房屋的售价等等。测试数据集有 1461 个观测值，每个数据集有 78 个特征。我们的任务是使用“training”数据集的模型来预测数据集的销售价格。首先，查看因变量即房价的分布，由于原始分布不满足线性回归中因变量符合正态分布的条件，所以我们选择进行平滑处理，即对其取 log，让其尽可

能符合高斯分布，如图一所示，左侧为平滑处理前数据分布情况，右侧为平滑处理后分布情况。

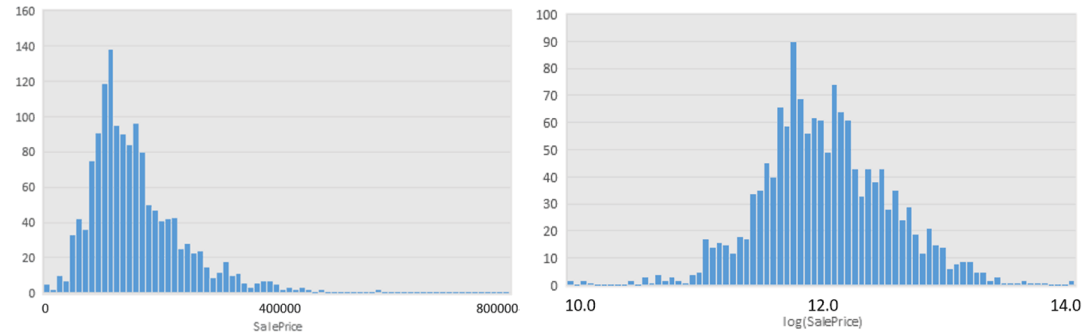
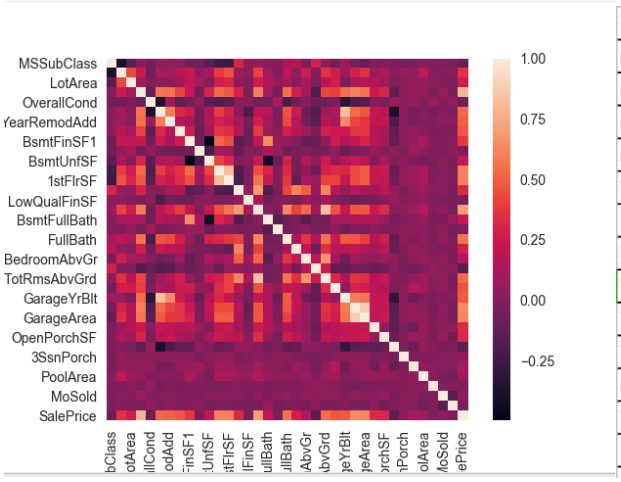


图 1

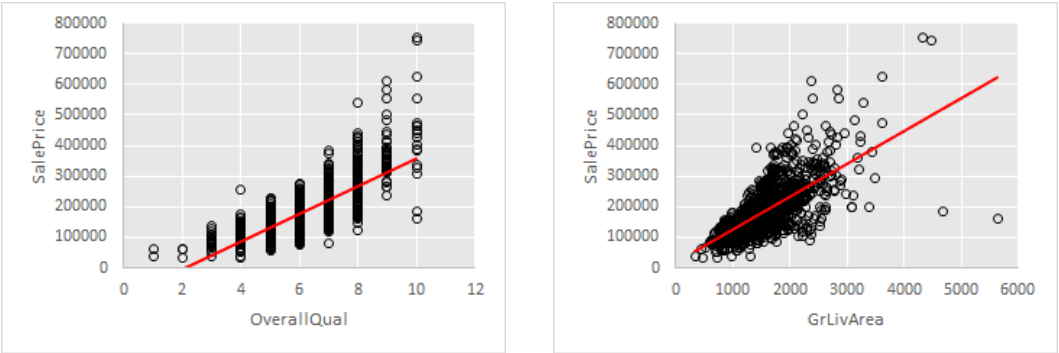
其次，图 2 为一些数值变量和房屋销售之间的相关系数矩阵和相关系数的正序排列。我们发现有几个变量与销售房价密切相关。我们下面将探索与图 3 中的销售价格高度相关的前六个变量。



变量	相关系数
OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
TotalBsmtSF	0.613581
1stFlrSF	0.605852
FullBath	0.560664
TotRmsAbvGrd	0.533723
YearBuilt	0.522897
YearRemodAdd	0.507101
GarageYrBlt	0.486362
MasVnrArea	0.477493
...	...

图 2

之后通过用 excel 对数据进行操作，得到变量与最终销售价格的相关系数。选取的相关性最大的六个变量的散点图与房价呈正相关，趋势曲线以红色表示，如图三所示。



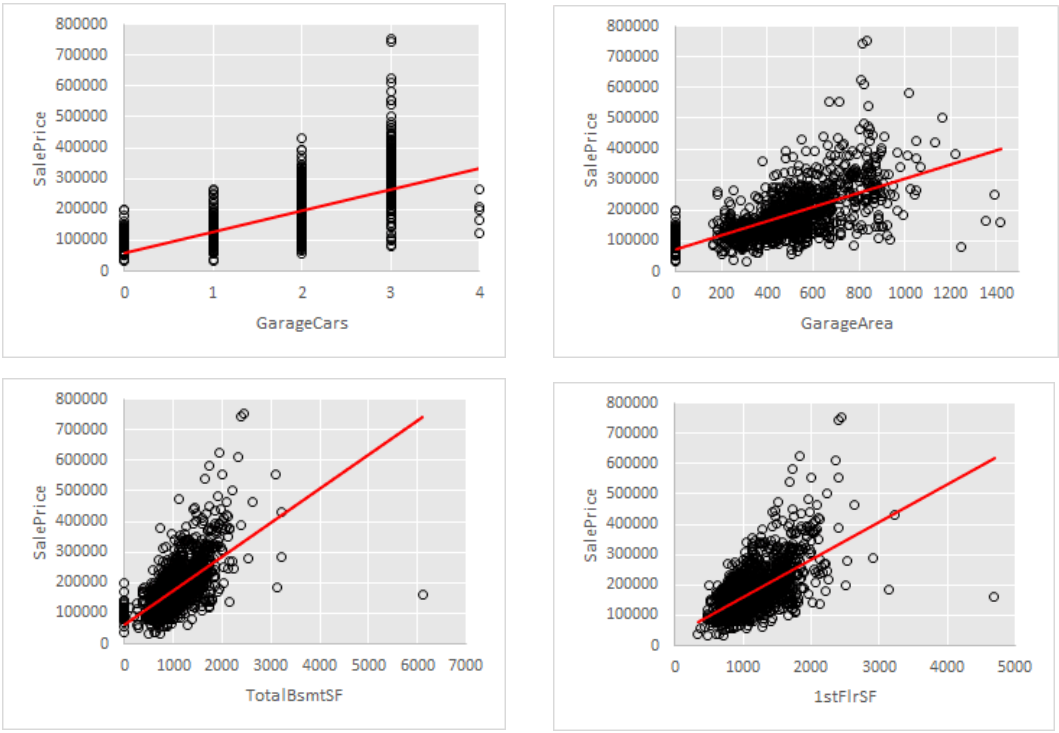
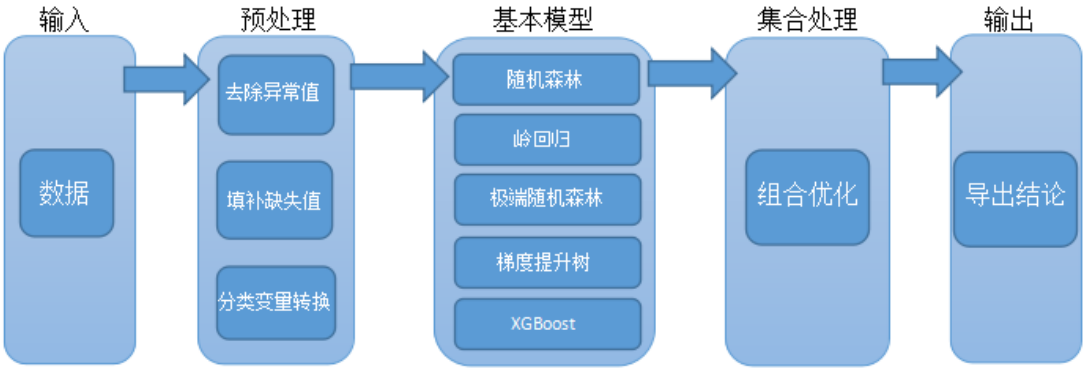


图 3

从中可以发现，有一些很明显的噪声点，在数据预处理中，我们将他们删除。因此，我们的整体思路可表示为下图所示的思维导图，细节将会在下面一一详述。



三．数据预处理

1. 所有变量中 Alley 变量有 1369 个缺失值，FireplaceQu 变量有 690 个缺失值，PoolQC 变量有 1453 个缺失值，Fence 变量有 1179 个缺失值，MiscFeature 变量有 1406 个缺失值。由于缺失值过多，我们将删掉这五个相关变量。

变量				变量			
PoolQC	7	non-null	object	BsmtExposure	1422	non-null	object
Fence	281	non-null	object	BsmtFinType1	1423	non-null	object
MiscFeature	54	non-null	object	BsmtFinType2	1422	non-null	object
Alley	91	non-null	object	Electrical	1459	non-null	object
FireplaceQu	770	non-null	object	GarageType	1379	non-null	object
LotFrontage	1201	non-null	float64	GarageYrBlt	1379	non-null	float64
MasVnrType	1452	non-null	object	GarageFinish	1379	non-null	object
MasVnrArea	1452	non-null	float64	GarageQual	1379	non-null	object
BsmtQual	1423	non-null	object	GarageCond	1379	non-null	object
BsmtCond	1423	non-null	object	MSSubClass	1460	non-null	int64
.....							

源代码为：

```
to_delete = ['Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature']
all_data = all_data.drop(to_delete, axis=1)
```

2. 然后对有缺失值的 14 个变量 LotFrontage，MasVnrType，MasVnrArea，BsmtQual，BsmtCond，BsmtExposure，BsmtFinTypel，BsmtFinType2，Electrical，GarageType，GarageYrBlt，GarageFinish，GarageQual，GarageCond，分别采用平均值填补。

3. 对于分类变量，我们采取将分类变量转换为虚拟变量或指示变量的方式。

源代码为：

```
all_data = pd.get_dummies(all_data)
all_data = all_data.fillna(all_data.mean())
```

4.对于 Sale Condition 变量，变量为 “Abnormal” 即在交易非正常情况下，我们将筛选出此类数据，删除并不予分析。

源代码为：

```
abnormal_idx = [3, 8, 12, 19, 38, 40, 46, 56, 88, 91 ,98 ,113 ,129 ,144 ,197 ,198 ,223 ,225 ,226 ,257 ,303 ,351 ,358,
                387 ,393 ,398 ,403 ,410 ,430 ,431 ,456 ,495 ,516 ,530 ,550 ,571 ,575 ,577 ,578 , 602 ,615 ,630 ,635,
                658 ,666 ,681 ,693 ,709 ,711 ,728 ,740 ,757 ,772 ,797 ,828 ,854 ,874 ,885 ,896 ,912 ,916 ,925 ,942,
                944 ,951 ,968 ,970 ,978 ,995 ,1001 ,1017 ,1024 ,1032 ,1049 ,1055, 1077 ,1080 ,1099 ,1108 ,1122 ,1131,
                1136 ,1140 ,1152 ,1182 ,1186 ,1200 ,1219 ,1220 ,1233 ,1234 ,1238 ,1245 ,1264 ,1279 ,
                1364 ,1366 ,1413 ,1428 ,1435 ,1449 ,1450]

train.drop(train.index[abnormal_idx], inplace=True)
```

5. 对离群值（Outlier）数据进行删除

源代码为：

```
train.drop(train[(train['GrLivArea'] > 4000)].index)
train.drop(train[(train['TotalBsmtSF'] > 6000)].index)
train.drop(train[(train['1stFlrSF'] > 4000)].index)
```

四．模型建立

总体思路：

运用五个基本的回归模型，分别为随机森林、岭回归、Extra trees（极端随机森林）、Gradient boosted tree、XGBoost；调整超参数进行模型训练，用 GridSearchCV 模块自动调参，获取自动交叉验证的最好分数，比较模型优劣。

其次，结合岭回归和随机森林的模型进行学习，产生合并模型。

同时获取模型预测的结果，将结果提交至 Kaggle 网站进行评价。

1.随机森林

随机森林采用多个决策树的投票机制来改善决策树，利用 Bootstraping 法，从原始数据集中有放回的抽样构造子数据集，利用子数据集构造子决策树，然后采用 Bagging 策略来获得最终结果（多数投票机制）。

- 优点：

  a） 对于多维特征的数据集分类有很高的效率；

  b） 可以做特征重要性的选择，运行效率和准确率较高，实现起来也比较简单；

- 缺点：

在数据噪音比较大的情况下会过拟合，且该缺点对结果影响比较大。

```
def model_random_forest(Xtrain, Xtest, ytrain):
    X_train = Xtrain
    y_train = ytrain
    rfr = RandomForestRegressor(n_jobs=1, random_state=0, max_depth=11)#选择n_jobs参量，结果最好

    param_grid = {'n_estimators': [300, 500], 'max_features': [10, 12, 14, 18]}
    model = GridSearchCV(estimator=rfr, param_grid=param_grid, n_jobs=1, cv=10, scoring=RMSE)
    model.fit(X_train, y_train)

    print('Random forest regression:')
    print('Best CV Score:')
    print(-model.best_score_)

    y_pred = model.predict(Xtest)
    return y_pred
```

2.岭回归

主要适用于过拟合严重或各变量之间存在多重共线性的时候，是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价获得回归系数更为符合实际、更可靠的回归方法，对病态数据的拟合要强于最小二乘法。

- 优点：

a）岭回归可以解决特征数量比样本量多的问题；

b）岭回归作为一种缩减算法可以判断哪些特征重要或者不重要，有点类似于降维的效果，保留更少的特征能够减少模型的复杂程度；

c）缩减算法可以看作是对一个模型增加偏差的同时减少方差；

- 缺点：

对于系数的估计是有偏的；

```
def model_ridge(Xtrain, Xtest, ytrain):
    X_train = Xtrain
    y_train = ytrain

    ridge = linear_model.Ridge()
    param_grid = {'alpha': np.logspace(-3, 2, 50)}
    model = GridSearchCV(estimator=ridge, param_grid=param_grid, n_jobs=1, cv=10, scoring=RMSE)
    model.fit(X_train, y_train)
    print('Ridge Regression:')
    print('Best CV score:')
    print(-model.best_score_)

    y_pred = model.predict(Xtest)
    return y_pred
```

3.Extra trees（极端随机森林）

该算法与随机森林算法十分相似，都是由许多决策树构成。但该算法与随机森林有两点主要的区别:

一是随机森林应用的是 Bagging 模型，而 ET 是使用所有的训练样本得到每棵决策树，也就是每棵决策树应用的是相同的全部训练样本；

二是随机森林是在一个随机子集内得到最佳分叉属性，而 ET 是完全随机的得到分叉值，从而实现对决策树进行分叉的；

- 优点：

a）最佳分叉属性是随机选择的，比随机森林的随机性更强，单棵决策树的预测结果往往不准确，但多棵决策树组合在一起，有很好的预测效果；

```
def model_extra_trees_regression(Xtrain, Xtest, ytrain):
    X_train = Xtrain
    y_train = ytrain

    etr = ExtraTreesRegressor(n_jobs=1, random_state=0,
            n_estimators=500, max_features=20)
    param_grid = {}
    model = GridSearchCV(estimator=etr, param_grid=param_grid, n_jobs=1, cv=10, scoring=RMSE)
    model.fit(X_train, y_train)
    print('Extra trees regression:')
    print('Best CV Score:')
    print(-model.best_score_)

    y_pred = model.predict(Xtest)
    return y_pred
```

4.Gradient boosted tree（梯度提升树）

GBT 是决策树的集合，迭代地训练决策树以便使损失函数最小化。

- 优点：

a）对混合数据的的天然处理能力，可以灵活处理各种类型的数据，包括连续值和离散值；

b）强大的预测能力（主要指算法本身的能力强大，一般性能好），在相对少的调参时间情况下，预测的准备率也可以比较高。这个是相对 SVM 来说的；

c）使用一些健壮的损失函数，对异常值的鲁棒性非常强；

- 缺点：

a）由于弱学习器之间存在依赖关系，难以并行训练数据；

```
def model_gradient_boosting_tree(Xtrain, Xtest, ytrain):
    X_train = Xtrain
    y_train = ytrain
    gbr = GradientBoostingRegressor( random_state=0, max_features=10,
            learning_rate=0.05, subsample=0.8)
    param_grid = {'max_depth': [6,7,8],'n_estimators': [500,800,1000]}
    model = GridSearchCV(estimator=gbr, param_grid=param_grid, n_jobs=1, cv=20, scoring=RMSE)
    model.fit(X_train, y_train)
    print('Gradient boosted tree regression:')
    print('Best CV Score:')
    print(-model.best_score_)

    y_pred = model.predict(Xtest)
    return y_pred
```

5.XGBoost（Extreme Gradient Boosting）

Gradient boosting 是 boosting 的其中一种方法，所谓 **Boosting** ，就是将弱分离器 f_i(x) 组合起来形成强分类器 F(x) 的一种方法；

- 优点：

a）对损失函数做了改进。传统 GBDT 在优化时只用到一阶导数信息，xgboost 则对代价函数进行了二阶泰勒展开；

b）可以防止过拟合。xgboost 在代价函数里加入了正则项，用于控制模型的复杂度；

```
def model_xgb_regression(Xtrain, Xtest, ytrain):
    X_train = Xtrain
    y_train = ytrain

    xgbreg = xgb.XGBRegressor( n_estimators=500, max_depth=7, eta = 0.05,
                              learning_rate=0.05, subsample=0.8, colsample_bytree=0.75)
    param_grid = {'seed': [0,500,1000]}
    model = GridSearchCV(estimator=xgbreg, param_grid=param_grid, n_jobs=1, cv=10, scoring=RMSE)
    model.fit(X_train, y_train)
    print('Extreme Gradient Boosting regression:')
    print('Best CV Score:')
    print(-model.best_score_)

    y_pred = model.predict(Xtest)
    return y_pred
```

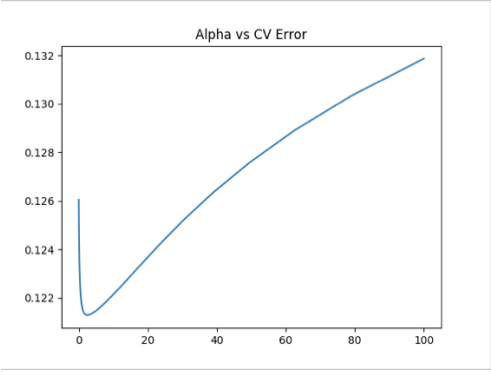
6. 基本模型比较

	优点	缺点
随机森林	实现简单；不容易过度拟合；能处理高纬度数据；平衡数据误差；可以选出重要特征	数据噪声过大时还是容易过度拟合；其随机性对于模型难以进行解释
岭回归	改良的OLS估计，在存在共线性问题和病态数据偏多的研究中有较大的实用价值。	结果是有偏的，降低了精度
Extra trees（极端随机森林）	随机性比随机森林更强，模型的方差相对于RF进一步减少，在某些时候，extra trees的泛化能力比RF更好。	但是偏倚相对于RF进一步增大
Gradient boosted tree（梯度提升树）	适用面广，几乎可用于所有回归问题（线性/非线性），亦可用于二分类问题	对异常值非常敏感
XGBoost（Extreme Gradient Boosting）	有效防止过度拟合；精度高；可适应纬度高的情况；	调参复杂

7. 模型合并

将岭回归与随机森林合并：

首先求出岭回归系数的使得 CV 值最大的 alpha 系数



模型代码：

```
def mode_ensemble_rfr_ridge(Xtrain, Xtest, ytrain):
    X_train = Xtrain
    y_train = ytrain
    X_test = Xtest

    ridge =  linear_model.Ridge(alpha=2.329952)
    rf = RandomForestRegressor(n_estimators=500, max_features=.3)
    ridge.fit(X_train, y_train)
    rf.fit(X_train, y_train)
    y_ridge = np.expm1(ridge.predict(X_test))
    y_rf = np.expm1(rf.predict(X_test))

    y_pred = (y_ridge+y_rf)/2
    return y_pred
```

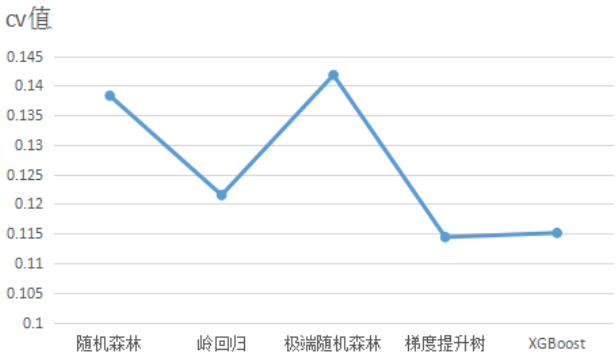
六．结果分析与评价

1. 基本结果分析

五个基本模型的比较：

（比较 cross-validation score 的负数），数值越小，回归效果越好。

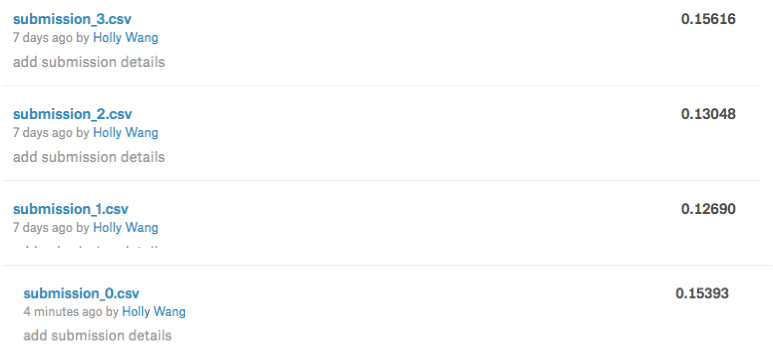
```
Random forest regression:
Best CV Score:
0.13825001670785134
Gradient boosted tree regression:
Best CV Score:
0.11444568720434202
Extreme Gradient Boosting regression:
Best CV Score:
0.1150576611704134
Extra trees regression:
Best CV Score:
0.14182317000239522
Ridge Regression:
Best CV score:
0.12154299504355613
```



因为合并模型没有获取 CV score，所以用 Kaggle 评价结果。

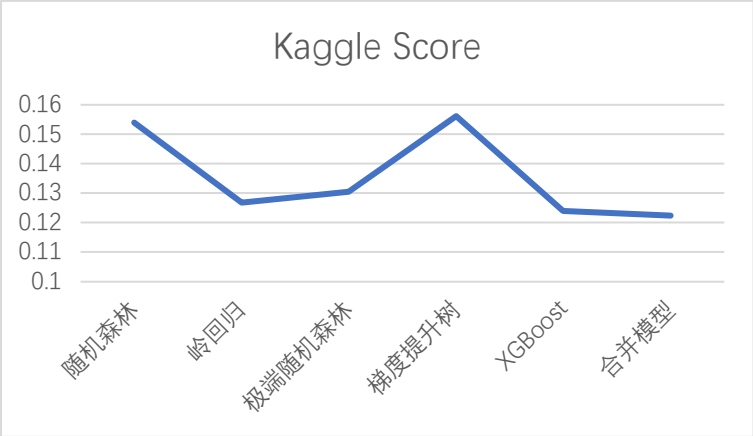
以下是 Kaggle 结果：

数字代表模型顺序，顺序按照以上介绍。



submission_4.csv 0.12405
a few seconds ago by Holly Wang
add submission details

submission5.csv 0.12242
a few seconds ago by Holly Wang
add submission details



比赛结果为排名 20%左右：

1069/5425
Top 20%

2. 分析与讨论

从 Kaggle 提供的结果来看，合并模型结果最优，其次是 XGBoost 模型。这与自己预估的结果相类似。回归模型基本有效。

反思：

1. 对参数的调整范围比较宽泛，没有精确估计。
2. 极端随机森林在 Kaggle 的模型评价比在交叉验证环节中要好；相反，梯度提升树的情况刚好与之相反，在实际评价中表现较差，可能还是存在 overfitting 的情况。
3. 变量分析不够细致，以后可以进行主成分分析。

团队贡献：

11611512 王丹妤 模型设计、所有代码以及报告的结果讨论部分

11510378 李莉欣 报告的模型特点整理、图部分

11510590 高宇馨 PPT 制作以及报告（概况、数据预处理等其他部分）