机器学习课程论文：分析房价

1 引言

预测房价一直是一个比较困难的事情，因为影响因素实在复杂。为了更好地理解各个因素在房价制定中起到的作用，本文将分析美国某房价数据库的资料，尝试理解不同因素所产生的不同效果。本文将使用随机森林回归器、支持向量机回归器、背景梯度提升回归器、自适应增强回归器等手段分析样本中变量与房价之间的关系。

2 分析过程

2.1 数据处理

经过统计，该训练集除房价外共有 80 个变量。其中有这些文本形式的变量：

```
#cat_attribute
cat_att = ["MSZoning", "Street", "Alley", "LotShape", "
    LandContour", "Utilities", "LotConfig", "LandSlope", "
    Neighborhood", "Condition1", "Condition2", "BldgType", "
    HouseStyle", "RoofStyle", "RoofMatl", "Exterior1st", "
    Exterior2nd", "MasVnrType", "ExterQual", "ExterCond", "
    Foundation", "BsmtQual", "BsmtCond", "BsmtExposure", "
    BsmtFinType1", "BsmtFinType2", "Heating", "HeatingQC", "
    CentralAir", "Electrical", "KitchenQual", "Functional", "
    FireplaceQu", "GarageType", "GarageFinish", "GarageQual",
    "GarageCond", "PavedDrive", "PoolQC", "Fence", "
    MiscFeature", "SaleType", "SaleCondition"]
```

在逐一绘制这些变量与房价的散点图后，本文认为有一些影响较小的因素可不予考虑：

```
#影响较小的因素不予考虑
cat_ignore = [" Utilities ", "PoolQC"]
num_ignore = ["ScreenPorch", "PoolArea", "MiscVal", "YrSold",
    "BsmtUnfSF", "2ndFlrSF", "LotArea", "LotFrontage", "
    MasVnrArea", "WoodDeckSF"]
```

在运用复合了 SimpleImputer 和 StandardScaler 的 Pipeline 处理完数字数据，并且用 OneHotEncoder 处理完文本数据后；训练集一共有 295 个特征值。其中前 27 个是数字特

征值，后 268 个是文本特征值。在我们运用随机森林回归器计算特征的重要程度后，图 1（以及附录 A.1）显示了特征的重要程度的分布情况。

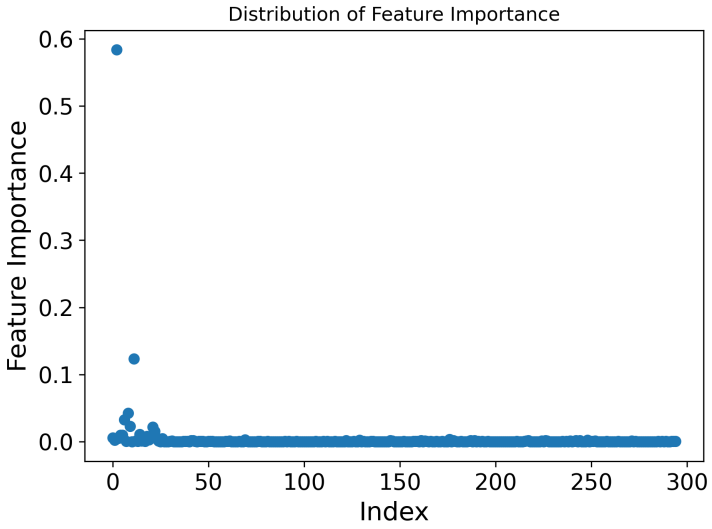
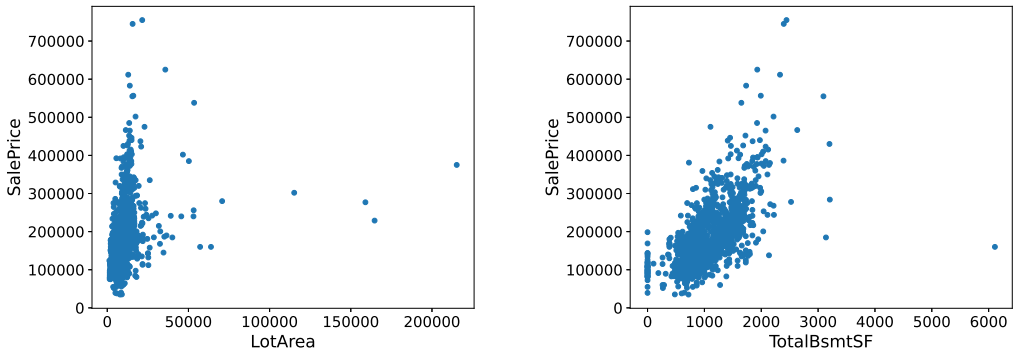


图 1: Feature Importance

从附录 A.1 中我们可以知道有两个影响很大的特征：第三个特征 LotArea（房屋总面积）具有 58.36% 的影响力；第 12 个特征 TotalBsmtSF（地下室面积）具有 12.34% 的影响力。图 2(a) 和图 2(b) 显示了这两个特征与房价的关系。由于仅有 8 个特征具有超过 1% 的影响力，而其他特征的影响力非常低；因此我选出前 100 个最有影响力的特征（合计 98.65% 的影响力）并用 PCA 在保留 98% 的方差解释率的条件下对它们进行降维。最后用于训练的特征大约有 96%（0.98^2）的影响力。

此外，为了得到更好的训练效果，我用 StandardScaler 对房价进行了归一化。



(a) Lot Size in square feet

(b) Total square feet of basement area

图 2: 最重要的两个特征

2.2 模型训练和微调

我使用了三种模型训练器：支持向量机回归器（svm.SVR）、背景梯度提升回归器（GradientBoostingRegressor）和自适应增强回归器（AdaBoostRegressor）。针对每一种训练器，

先随机生成超参数的集合，再用网格搜索（GridSearchCV）从中选出最好的模型。最后依靠投票回归器（VotingRegressor）集成所有优秀的回归器。

在训练的结束阶段，我集成了 16 个支持向量机回归器、16 个背景梯度提升回归器和 10 个自适应增强回归器。支持向量机回归器（见附录 B.1）的准确度大约为 90%，背景梯度提升回归器（见附录 B.2）的准确度大约为 97%，自适应增强回归器（见附录 B.3）的准确度大约为 88%。在集成了这 42 个回归器后，投票回归器能拿到 95.32% 的成绩。

3 结果评估

从以上分析中我们可以看出：房屋的面积在很大程度上决定了房价，房屋面积与地下室面积一共有 70% 左右的重要性。这符合常识。

此外，我一共使用了三种基础模型：支持向量机回归器、背景梯度提升回归器和自适应增强回归器。关于 SVR，使用了“rgb”核与“linear”核的向量机具有更好的拟合效果。通过比较背景梯度提升回归器和自适应增强回归器的性能，我们发现：在房价拟合方面，修饰残差（背景梯度提升回归器）比修改权重（自适应增强回归器）会取得更好的效果。

不得不感叹：买房一直以来都是一件麻烦事。

A 附录：数据处理

A.1 特征重要性

本表列举前 10 个最重要的特征的索引及其重要程度：

索引	特征重要性
2	0.583582266714989
11	0.123422778042043
8	0.04267767142987
6	0.032897373099546
9	0.022698788009794
21	0.021788262646192
22	0.015713335639012
14	0.010633035973981
4	0.009801764735621
5	0.009671165316304

表 1: 最重要的 10 个特征

B 附录：模型训练和微调

B.1 支持向量机回归器

从以下结果可以看出，支持向量机回归器在训练集上的拟合程度大概能达到 90% 左右的好成绩。

```
SVR: 1/16 done! score: 0.933357640391794 SVR(C=1.4, degree=2, epsilon=0.3)
SVR: 2/16 done! score: 0.812993079038421 SVR(C=0.4, degree=10, epsilon=0.7, kernel='linear')
SVR: 3/16 done! score: 0.933357640391794 SVR(C=1.4, degree=8, epsilon=0.3)
SVR: 4/16 done! score: 0.933357640391794 SVR(C=1.4, degree=9, epsilon=0.3)
SVR: 5/16 done! score: 0.8397347101193189 SVR(C=0.4, epsilon=0.4, kernel='linear')
SVR: 6/16 done! score: 0.9392571797173856 SVR(C=1.2, degree=10, epsilon=0.2)
SVR: 7/16 done! score: 0.918317394806964 SVR(C=1.4, degree=6, epsilon=0.4)
SVR: 8/16 done! score: 0.8826670168229886 SVR(C=1.6, degree=8, epsilon=0.6)
SVR: 9/16 done! score: 0.9232008201349666 SVR(C=1.6, degree=6, epsilon=0.4)
SVR: 10/16 done! score: 0.8270309681302039 SVR(C=0.2, degree=5, epsilon=0.6, kernel='linear')
SVR: 11/16 done! score: 0.8275590083595 SVR(C=0.6, degree=9, epsilon=0.6, kernel='linear')
SVR: 12/16 done! score: 0.8999157985523356 SVR(C=1.4, degree=8, epsilon=0.5)
SVR: 13/16 done! score: 0.9464388352892089 SVR(C=1.4, degree=8, epsilon=0.2)
SVR: 14/16 done! score: 0.8602295191998717 SVR(C=1.6, degree=2, epsilon=0.7)
SVR: 15/16 done! score: 0.9044078921223938 SVR(C=1.6, epsilon=0.5)
SVR: 16/16 done! score: 0.9392571797173856 SVR(C=1.2, epsilon=0.2)
```

B.2 背景梯度提升回归器

从以下结果可以看出，背景梯度提升回归器在训练集上的拟合程度大概能达到 97% 左右的好成绩。

```
GBR: 1/16 done! score: 0.9908829058404207 GradientBoostingRegressor(learning_rate=0.5, loss='huber', max_features='auto')
GBR: 2/16 done! score: 0.9835105756566681
```

```
    GradientBoostingRegressor(criterion='mse', learning_rate
    =0.3, loss='huber', max_features='auto')
GBR: 3/16 done! score: 0.981591018845118
    GradientBoostingRegressor(criterion='mse', learning_rate
    =0.3, loss='huber', max_features='auto')
GBR: 4/16 done! score: 0.9281499579931975
    GradientBoostingRegressor(learning_rate=0.4, loss='lad',
    max_features='sqrt')
GBR: 5/16 done! score: 0.9510026534127038
    GradientBoostingRegressor(criterion='mse', loss='huber',
    max_features='auto')
GBR: 6/16 done! score: 0.9744864689368227
    GradientBoostingRegressor(learning_rate=0.2, loss='huber',
    max_features='auto')
GBR: 7/16 done! score: 0.966326575399972
    GradientBoostingRegressor(max_features='auto')
GBR: 8/16 done! score: 0.9679114534051618
    GradientBoostingRegressor(learning_rate=0.2, max_features
    ='sqrt')
GBR: 9/16 done! score: 0.9723936496537421
    GradientBoostingRegressor(criterion='mse', learning_rate
    =0.3, loss='huber', max_features='sqrt')
GBR: 10/16 done! score: 0.9736056113189555
    GradientBoostingRegressor(learning_rate=0.2, loss='huber',
    max_features='auto')
GBR: 11/16 done! score: 0.9749603276526679
    GradientBoostingRegressor(criterion='mse', learning_rate
    =0.2, loss='huber', max_features='auto')
GBR: 12/16 done! score: 0.9484783064779051
    GradientBoostingRegressor(criterion='mse', max_features='
    sqrt')
GBR: 13/16 done! score: 0.9529577183211357
    GradientBoostingRegressor(criterion='mse', loss='huber',
    max_features='auto')
GBR: 14/16 done! score: 0.9736056113189555
    GradientBoostingRegressor(criterion='mse', learning_rate
    =0.2, loss='huber', max_features='auto')
GBR: 15/16 done! score: 0.9824417477535988
    GradientBoostingRegressor(learning_rate=0.2, max_features
    ='auto')
GBR: 16/16 done! score: 0.9748620191181754
    GradientBoostingRegressor(learning_rate=0.2, loss='huber',
```

```
        max_features='auto')
```

B.3 自适应增强回归器

从以下结果可以看出，自适应增强回归器的效果相对较差。它在训练集上的拟合程度大概能达到 88% 左右的准确度。

```
1  ABR: 1/10 done! score: 0.8920755524524758 AdaBoostRegressor(
       learning_rate=0.8, loss='square', n_estimators=80)
2  ABR: 2/10 done! score: 0.8838390677742329 AdaBoostRegressor(
       learning_rate=0.9, n_estimators=80)
3  ABR: 3/10 done! score: 0.8915948933101925 AdaBoostRegressor(
       learning_rate=1.2, loss='square', n_estimators=70)
4  ABR: 4/10 done! score: 0.883417704230598 AdaBoostRegressor(
       learning_rate=0.4, loss='exponential', n_estimators=90)
5  ABR: 5/10 done! score: 0.8891242646226272 AdaBoostRegressor(
       learning_rate=1.1, loss='exponential')
6  ABR: 6/10 done! score: 0.8857303626428468 AdaBoostRegressor(
       learning_rate=0.6, n_estimators=70)
7  ABR: 7/10 done! score: 0.882888944752859 AdaBoostRegressor(
       learning_rate=1.2, n_estimators=90)
8  ABR: 8/10 done! score: 0.8837334720348414 AdaBoostRegressor(
       learning_rate=0.8, loss='exponential', n_estimators=30)
9  ABR: 9/10 done! score: 0.8764709483237575 AdaBoostRegressor(
       n_estimators=40)
10 ABR: 10/10 done! score: 0.885898953214578 AdaBoostRegressor(
       learning_rate=1.1, n_estimators=80)
```