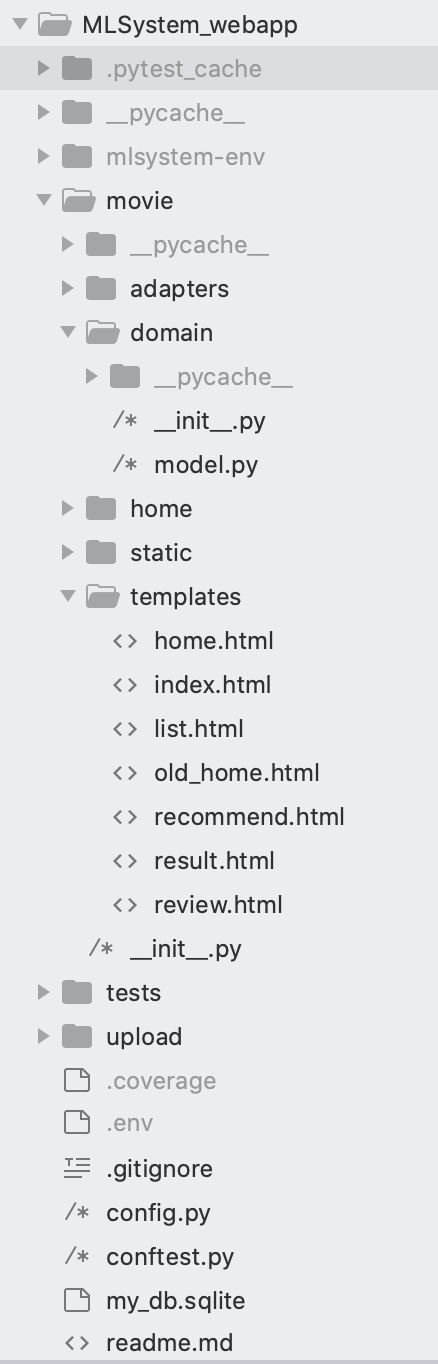
a11398\_a9364\_MLSystem

Task A House Price Prediction

工程整体架构

前后端架构:

Bootstrap，来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的，它简洁灵活，使得 Web 开发更加快捷

Flask 是一个微型的 Python 开发的 Web 框架，基于Werkzeug WSGI工具箱和Jinja2 模板引擎。 Flask使用BSD授权。 Flask也被称为“microframework”，因为它使用简单的核心，用extension增加其他功能。Flask没有默认使用的数据库、窗体验证工具。然而，Flask保留了扩增的弹性，可以用Flask-extension加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）。 它封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互

具体关键代码介绍：

forms（表单）（主要在template各种html）：存放表单对象

models（模型 ) (主要在domain/model.py）：存放数据模型，即库表在程序中的映射对象，以及对象之间的关系

routes（路由）：存放请求路由以及处理逻辑

static（静态文件）：Flask 约定存放静态文件的目录

templates（模板）：Flask 约定存放页面模板的目录

services（服务）：存放业务逻辑或者其他服务类功能

\_\_init\_\_.py：Flask app 初始化方法

config.py：项目配置文件

README.md - 仓库的说明，比如该项目的介绍等

wsgi.py - 项目的主要功能实现

config.py - 项目的配置

manage.py（非具体文件，你可以在命令行下使用） - 基于Flask-Script扩展的命令行脚本

requirements - 该项目所依赖的第三方包

unit\_tests - 单元测试相关代码

常用扩展（**Extensions**）：

**Flask**作为微框架（**microframework**），在开发过程中会经常使用各种扩展包。以下是一些常用扩展包的简介。

**Flask-SQLAlchemy -** 封装了**SQLAlchemy**，提供**ORM**

**Flask-Migrate -** 处理**SQLAlchemy**数据库的迁移（**migrations**）

**Flask-Script -** 支持在**Flask**里编写额外的脚本

**Flask-Bootstrap -** 封装了**Bootstrap**框架

**Flask-Login -** 提供账号**session**管理

**Flask-WTF -** 封装了**WTForms**，提供表单功能

**Flask-RESTful -** 提供快速构建**RESTAPIs**的能力

具体机器学习案例

**Business origin introduction**:

当我们拥有房屋属性数据和销售价格数据时，如果我们可以相互关联，这对于房地产网站和房屋购买者来说非常重要。

此外，该模型还可以更有效地帮助推荐系统发挥作用：知道预测价格对于价格敏感群体的房屋推荐非常重要。

我们有一个数据集，数据集中的每一行都描述了房屋的特征。我们的目标是根据房屋的特征预测销售价格。

评估模型基于模型预测的销售价格与实际销售价格之间的均方根误差（RMSE）。将RMSE转换为对数刻度，以确保预测昂贵房屋和廉价房屋的误差具有相同的得分影响。模型：每个交叉验证都适合许多模型（包括套索，山脊，svr，ker，ela，bay等）。所有训练后的模型在不同程度上过拟合训练数据。因此，为了做出最终的预测，我通过使用Stacking将它们的预测混合在一起以获得更可靠的预测。

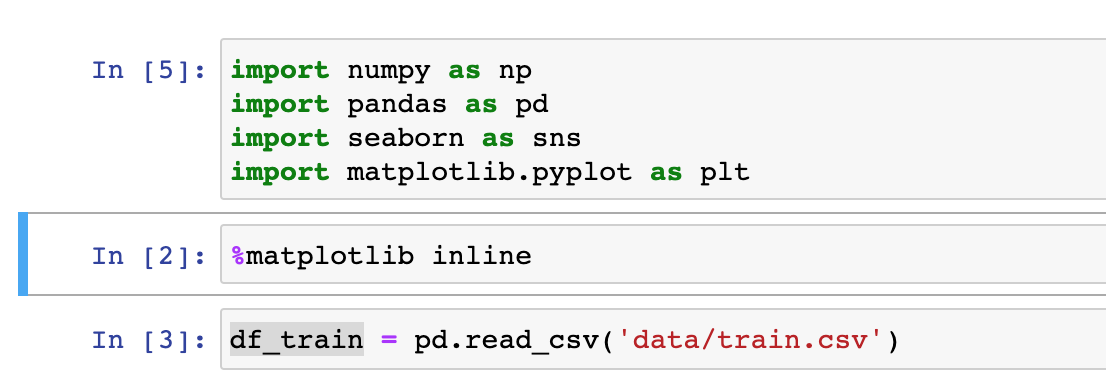
为了提高RMSE，我使用以下方法：

特征工程：主要为离散变量，特征组合和PCA分配值

模型融合：主要是加权平均和叠加

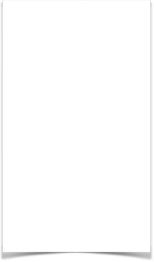
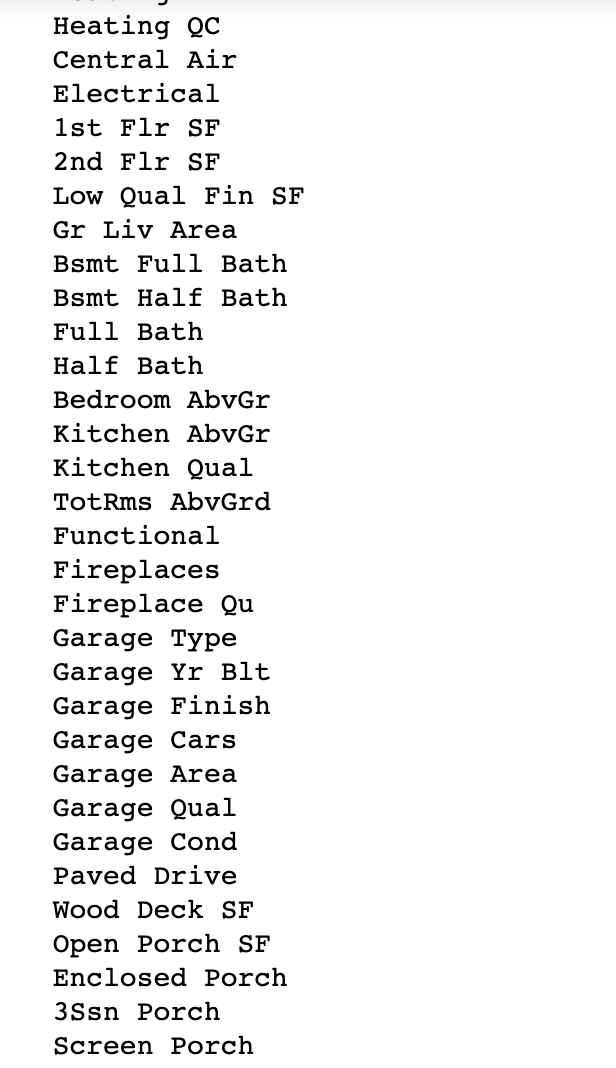
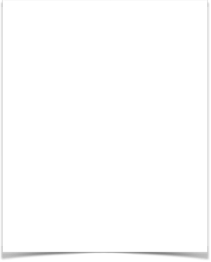
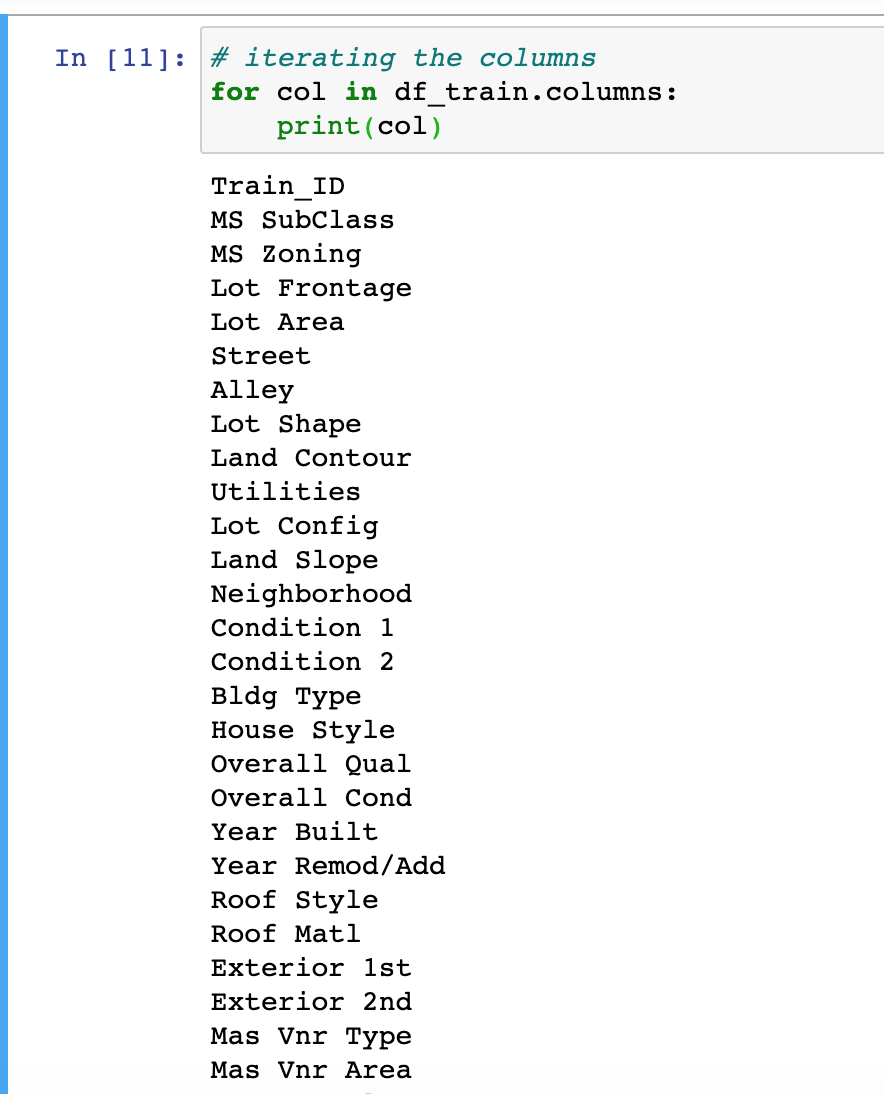
**Import necessary libraries**

主要使用： numpy,pandas, matplotlib.pyplot, seaborn… this libs to explore What attributes and data are needed for our predict.

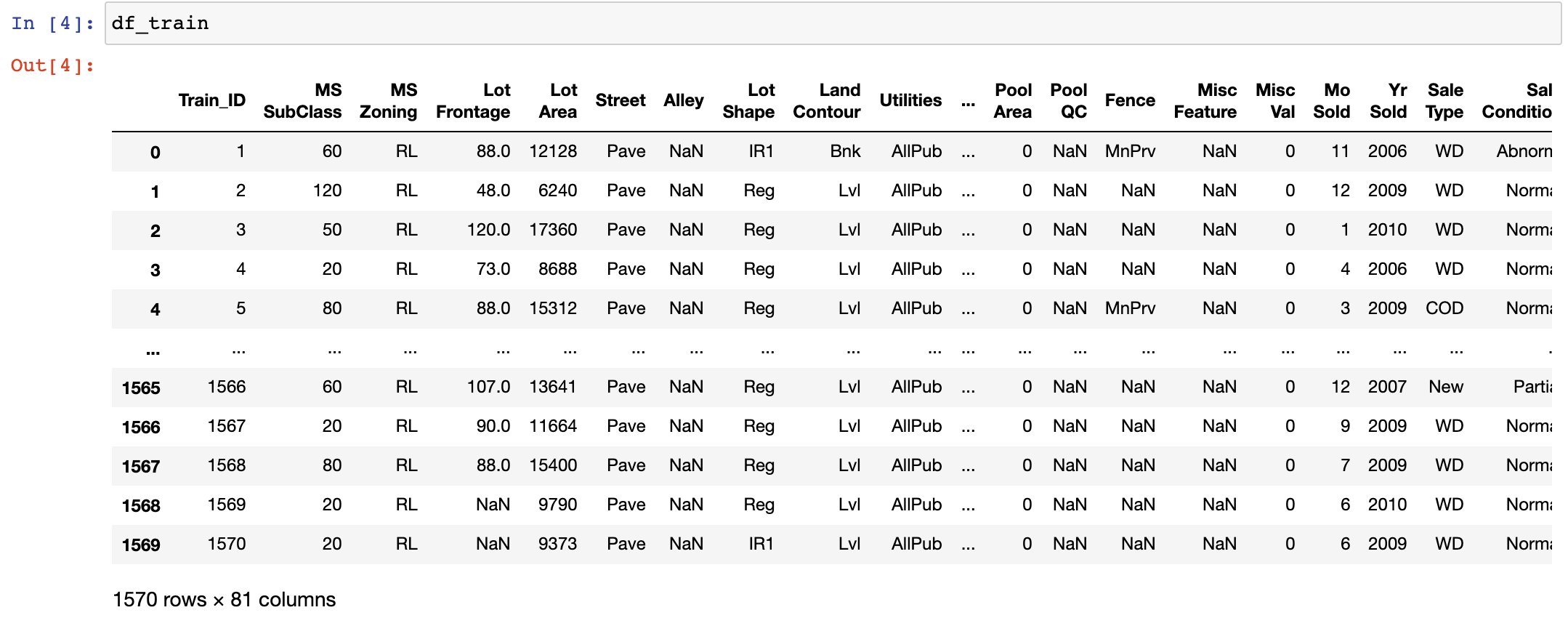


**Import Data**

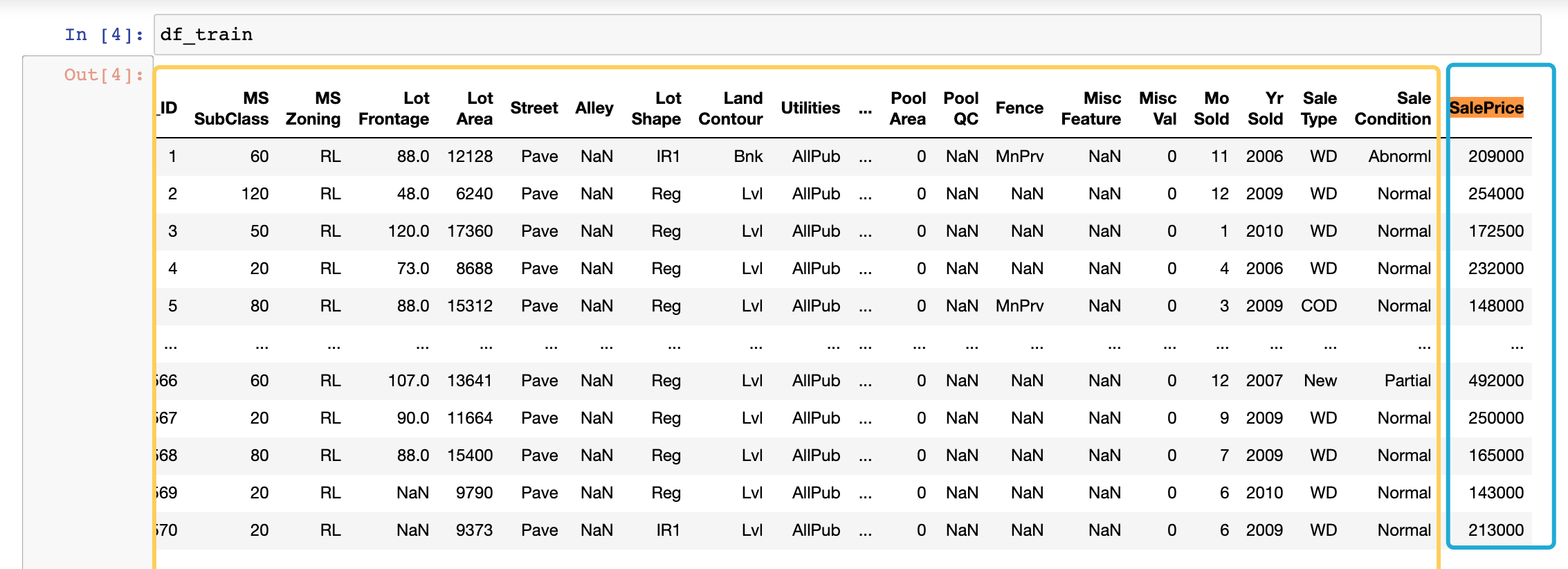
We explore all column names , read Kaggle\_House\_Data\_Description’s details and explainations.



Get all column names



Get data shape and some records

快速查看一下81列的值和列名，我们可以知道有80个要素，其中包括1个预测标签和79个要素标签。 具体分类如下 

仔细读取所有列名称作为特征以及列数据的解释。

For example:

MSSubClass: Identifies the type of dwelling involved in the sale.

MSZoning: Identifies the general zoning classification of the sale.

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Alley: Type of alley access to property

…

Fence: Fence quality

MiscFeature: Miscellaneous feature not covered in other categories

MiscVal: $Value of miscellaneous feature

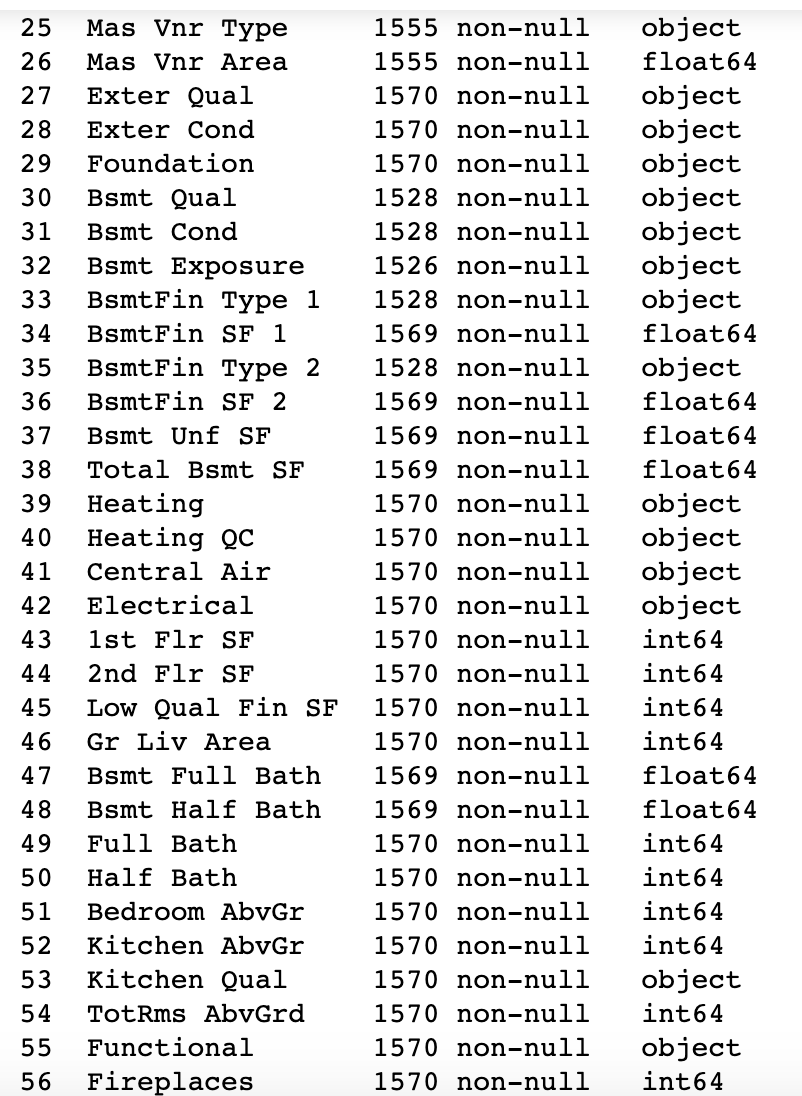
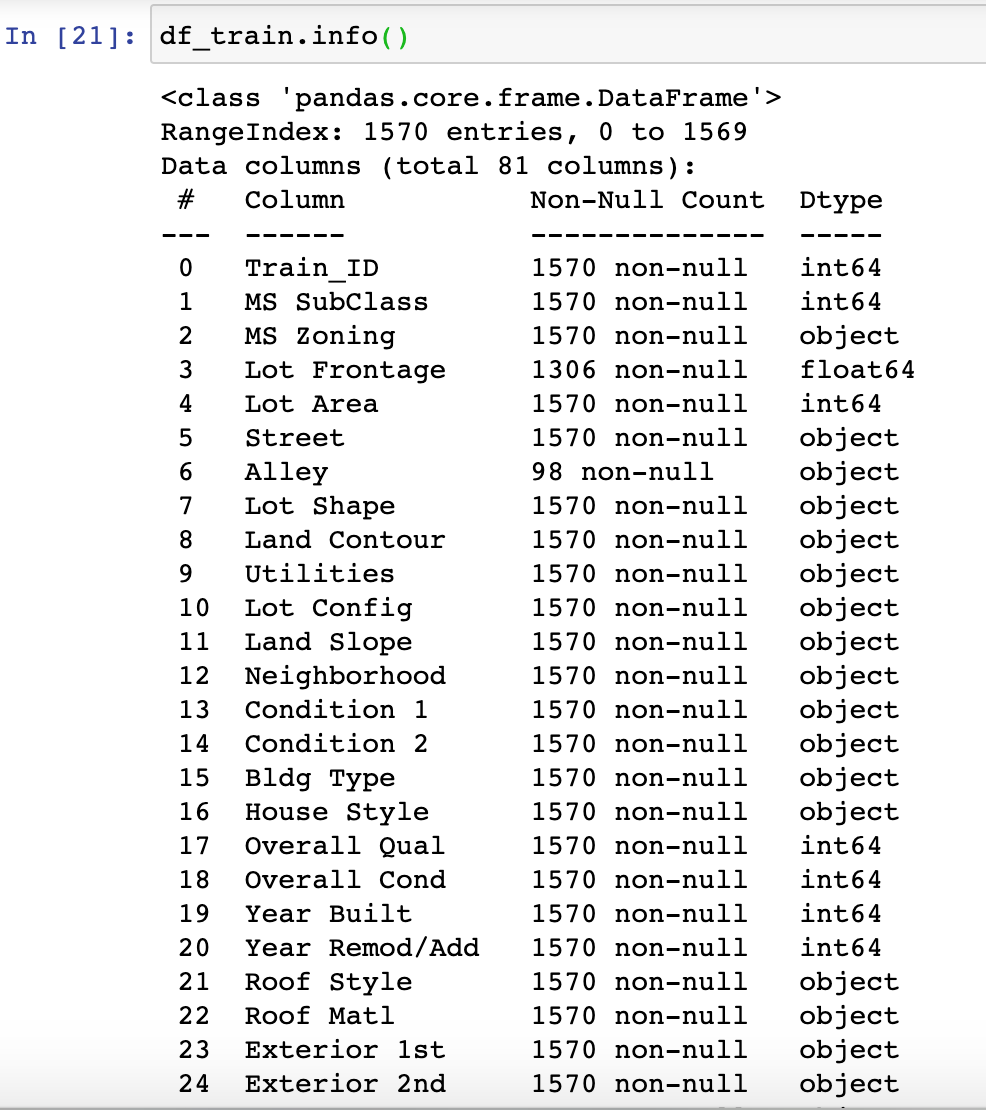
MoSold: Month Sold (MM)

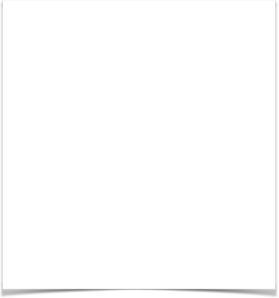
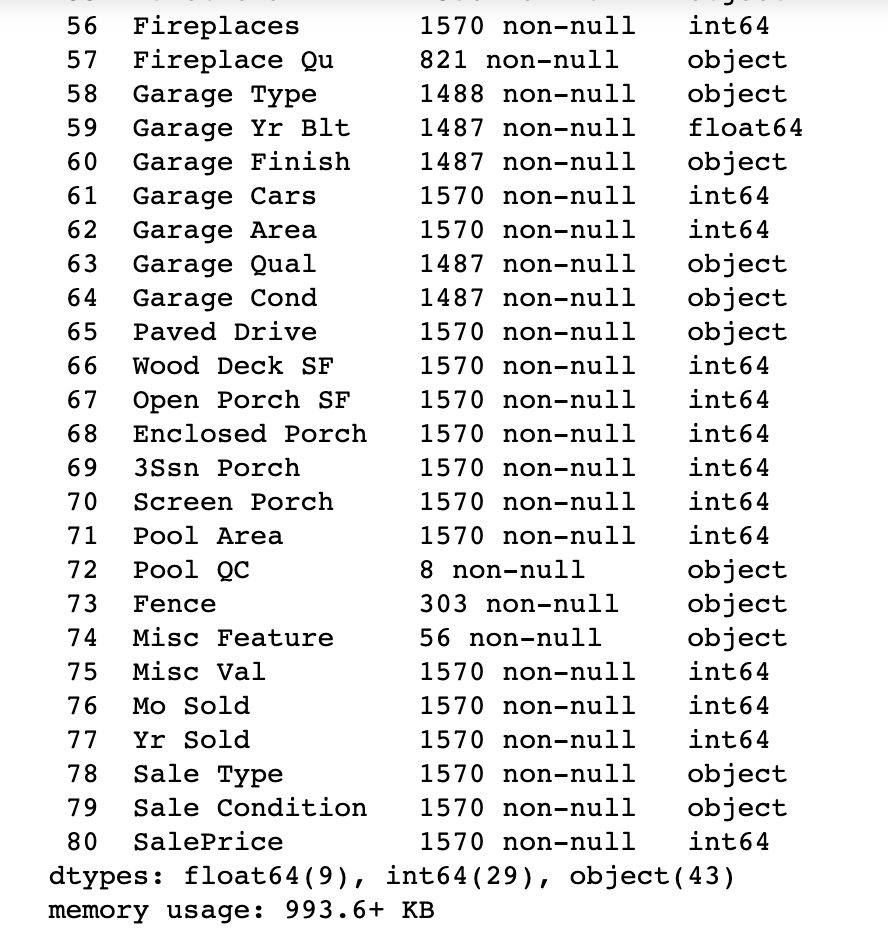
YrSold: Year Sold (YYYY)

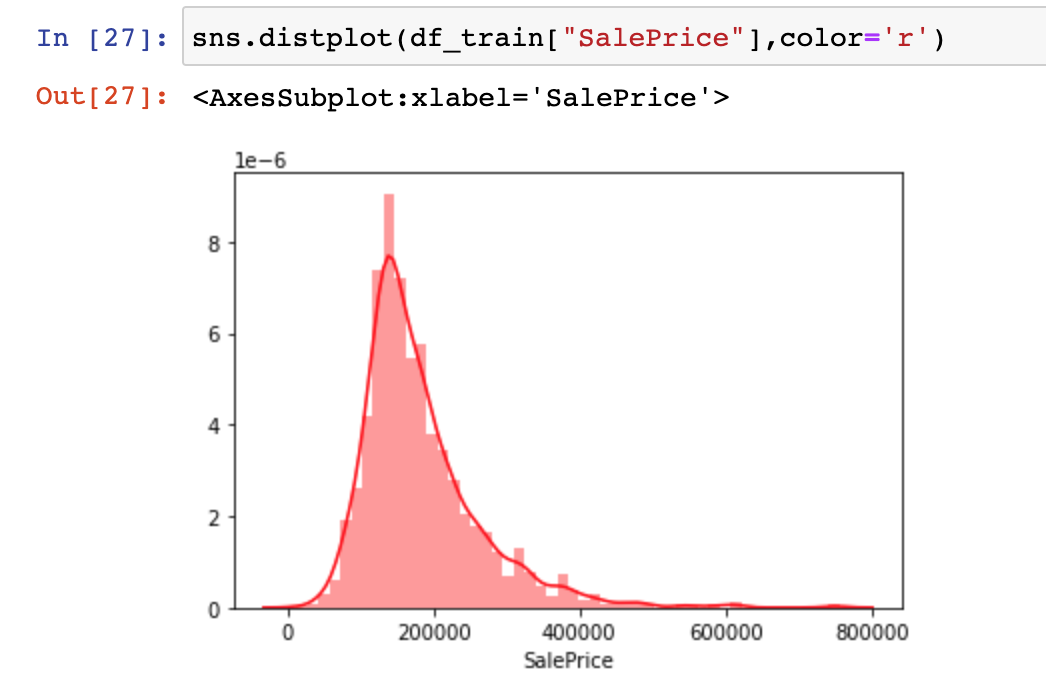
SaleType: Type of sale

SaleCondition: Condition of sale

**Look at the statistics**

查看数据集中每个数据列的默认数据，并查看哪些列的数据太少而需要删除 



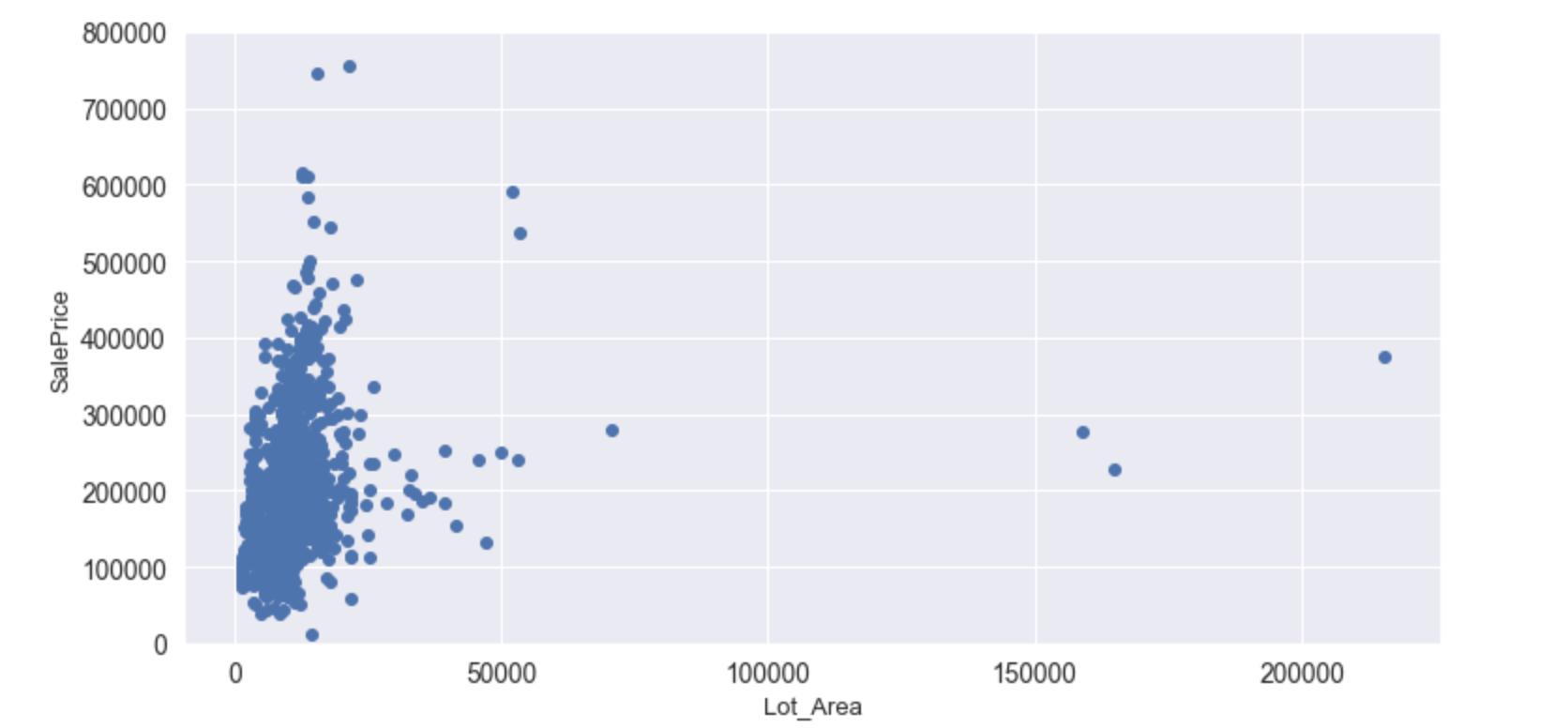
Through visualization, analyze the statistical law of the SalePrice itself

通过查看SalesPrice列的分布，我们发现大多数房屋价格都在$ 400,000之内，并且数据分布略向左

我们需要找到与salesprice最相关的列，我们进行测试，首先找到前20名，然后是前10名

我们可以看到20列的相关热图过多，因此我们将其分为10列，如下所示:

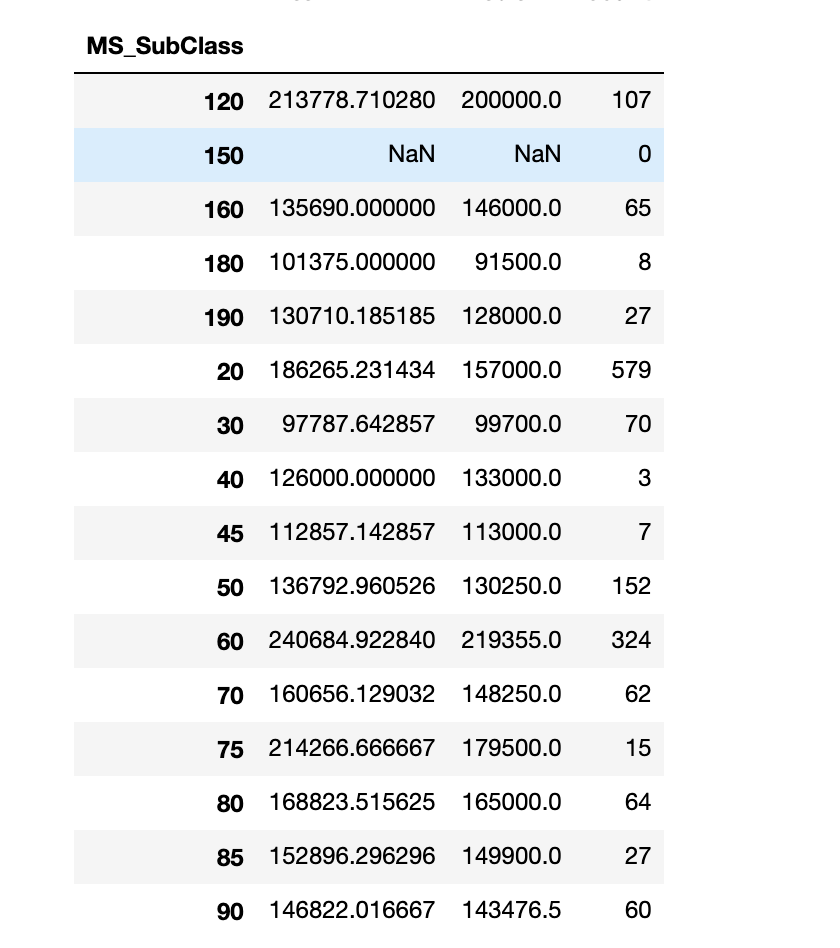


销售价格越相关，颜色越浅，通过观察：从图中可以看出 Overall Qual, Gr Live Area, Total Bsmt SF, 1st Flr SF, Garage Cars. 相关系数都比较高.

When we do visualization of price and columns we want to observe, we can find outlier points : right-bottom-corner points are not wanted. so we use :

train.drop(train[(train[“Lot\_Area"]>50000)&(train["SalePrice"]<400000)].index,inplace=True) to filter outlier data out.

**Feature Engineering**

或离散要素，通常使用pandas中的get\_dummies进行数字化，但这在Kaggle竞赛中可能还不够，因此我在下面使用的方法是按要素分组并计算要素的每个值的SalePrice平均值和中位数 ，然后以此为基础对值进行排序和分配，例如：:

Take MSSubClass as example : the feature of MSSubClass represents the type of house, and the data is grouped by it.

So We do the mapping as following:

MS\_SubClass.map({'180':1,

'30':2, '45':2,

'190':3, '50':3, '90':3,

'85':4, '40':4, '160':4,

'70':5, '20':5, '75':5, '80':5, '150':5,

'120': 6, '60':6})

Similar to MS\_SubClass.map, a total of more than 20 features are roughly ranked, as following:

def map\_into\_new\_values():

full["oMSSubClass"] = full.MS\_SubClass.map({'180':1,

'30':2, '45':2,

'190':3, '50':3, '90':3,

'85':4, '40':4, '160':4,

'70':5, '20':5, '75':5, '80':5, '150':5,

'120': 6, '60':6})

full["oMSZoning"] = full.MS\_Zoning.map({'C (all)':1, 'RH':2, 'RM':2, 'RL':3, 'FV':4})

full["oNeighborhood"] = full.Neighborhood.map({'MeadowV':1,

'IDOTRR':2, 'BrDale':2,

'OldTown':3, 'Edwards':3, 'BrkSide':3,

'Sawyer':4, 'Blueste':4, 'SWISU':4, 'NAmes':4,

'NPkVill':5, 'Mitchel':5,

'SawyerW':6, 'Gilbert':6, 'NWAmes':6,

'Blmngtn':7, 'CollgCr':7, 'ClearCr':7, 'Crawfor':7,

'Veenker':8, 'Somerst':8, 'Timber':8,

'StoneBr':9,

'NoRidge':10, 'NridgHt':10})

full["oCondition1"] = full.Condition\_1.map({'Artery':1,

'Feedr':2, 'RRAe':2,

'Norm':3, 'RRAn':3,

'PosN':4, 'RRNe':4,

'PosA':5 ,'RRNn':5})

full["oBldgType"] = full.Bldg\_Type.map({'2fmCon':1, 'Duplex':1, 'Twnhs':1, '1Fam':2, 'TwnhsE':2})

full["oHouseStyle"] = full.House\_Style.map({'1.5Unf':1,

'1.5Fin':2, '2.5Unf':2, 'SFoyer':2,

'1Story':3, 'SLvl':3,

'2Story':4, '2.5Fin':4})

full["oExterior1st"] = full.Exterior\_1st.map({'BrkComm':1,

'AsphShn':2, 'CBlock':2, 'AsbShng':2,

'WdShing':3, 'Wd Sdng':3, 'MetalSd':3, 'Stucco':3, 'HdBoard':3,

'BrkFace':4, 'Plywood':4,

'VinylSd':5,

'CemntBd':6,

'Stone':7, 'ImStucc':7})

full["oMasVnrType"] = full.Mas\_Vnr\_Type.map({'BrkCmn':1, 'None':1, 'BrkFace':2, 'Stone':3})

full["oExterQual"] = full.Exter\_Qual.map({'Fa':1, 'TA':2, 'Gd':3, 'Ex':4})

full["oFoundation"] = full.Foundation.map({'Slab':1,

'BrkTil':2, 'CBlock':2, 'Stone':2,

'Wood':3, 'PConc':4})

full["oBsmtQual"] = full.Bsmt\_Qual.map({'Fa':2, 'None':1, 'TA':3, 'Gd':4, 'Ex':5})

full["oBsmtExposure"] = full.Bsmt\_Exposure.map({'None':1, 'No':2, 'Av':3, 'Mn':3, 'Gd':4})

full["oHeating"] = full.Heating.map({'Floor':1, 'Grav':1, 'Wall':2, 'OthW':3, 'GasW':4, 'GasA':5})

full["oHeatingQC"] = full.Heating\_QC.map({'Po':1, 'Fa':2, 'TA':3, 'Gd':4, 'Ex':5})

full["oKitchenQual"] = full.Kitchen\_Qual.map({'Fa':1, 'TA':2, 'Gd':3, 'Ex':4})

full["oFunctional"] = full.Functional.map({'Maj2':1, 'Maj1':2, 'Min1':2, 'Min2':2, 'Mod':2, 'Sev':2, 'Typ':3})

full["oFireplaceQu"] = full.Fireplace\_Qu.map({'None':1, 'Po':1, 'Fa':2, 'TA':3, 'Gd':4, 'Ex':5})

full["oGarageType"] = full.Garage\_Type.map({'CarPort':1, 'None':1,

'Detchd':2,

'2Types':3, 'Basment':3,

'Attchd':4, 'BuiltIn':5})

full["oGarageFinish"] = full.Garage\_Finish.map({'None':1, 'Unf':2, 'RFn':3, 'Fin':4})

full["oPavedDrive"] = full.Paved\_Drive.map({'N':1, 'P':2, 'Y':3})

full["oSaleType"] = full.Sale\_Type.map({'COD':1, 'ConLD':1, 'ConLI':1, 'ConLw':1, 'Oth':1, 'WD':1,

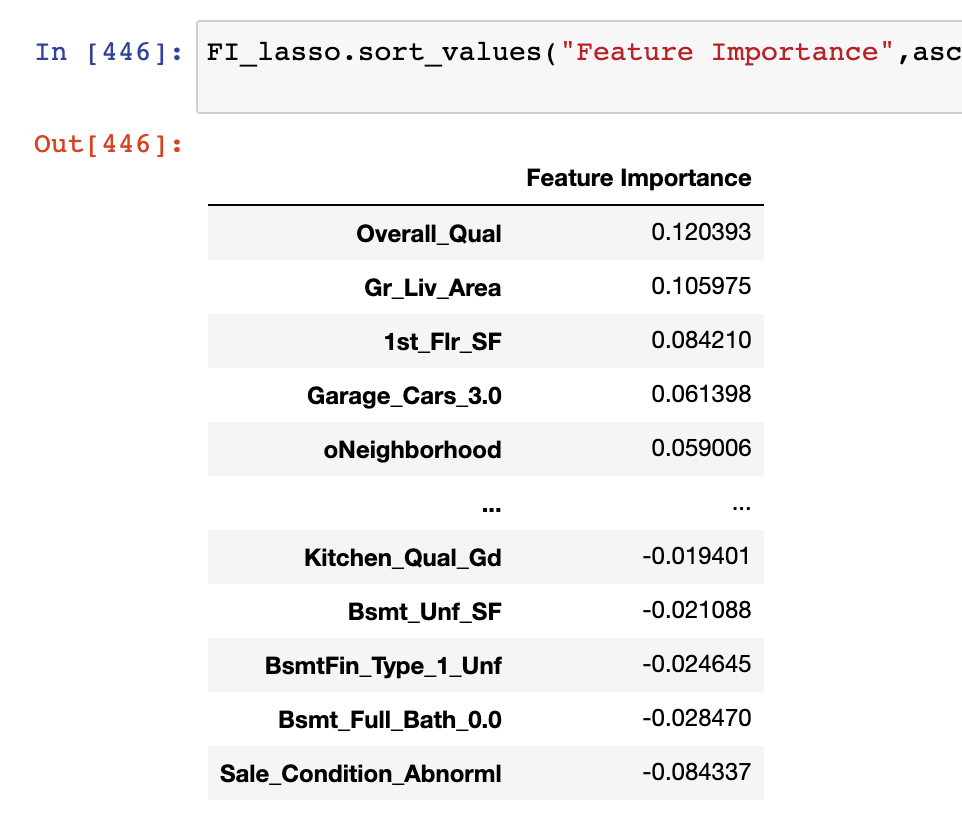
'CWD':2, 'Con':3, 'New':3})

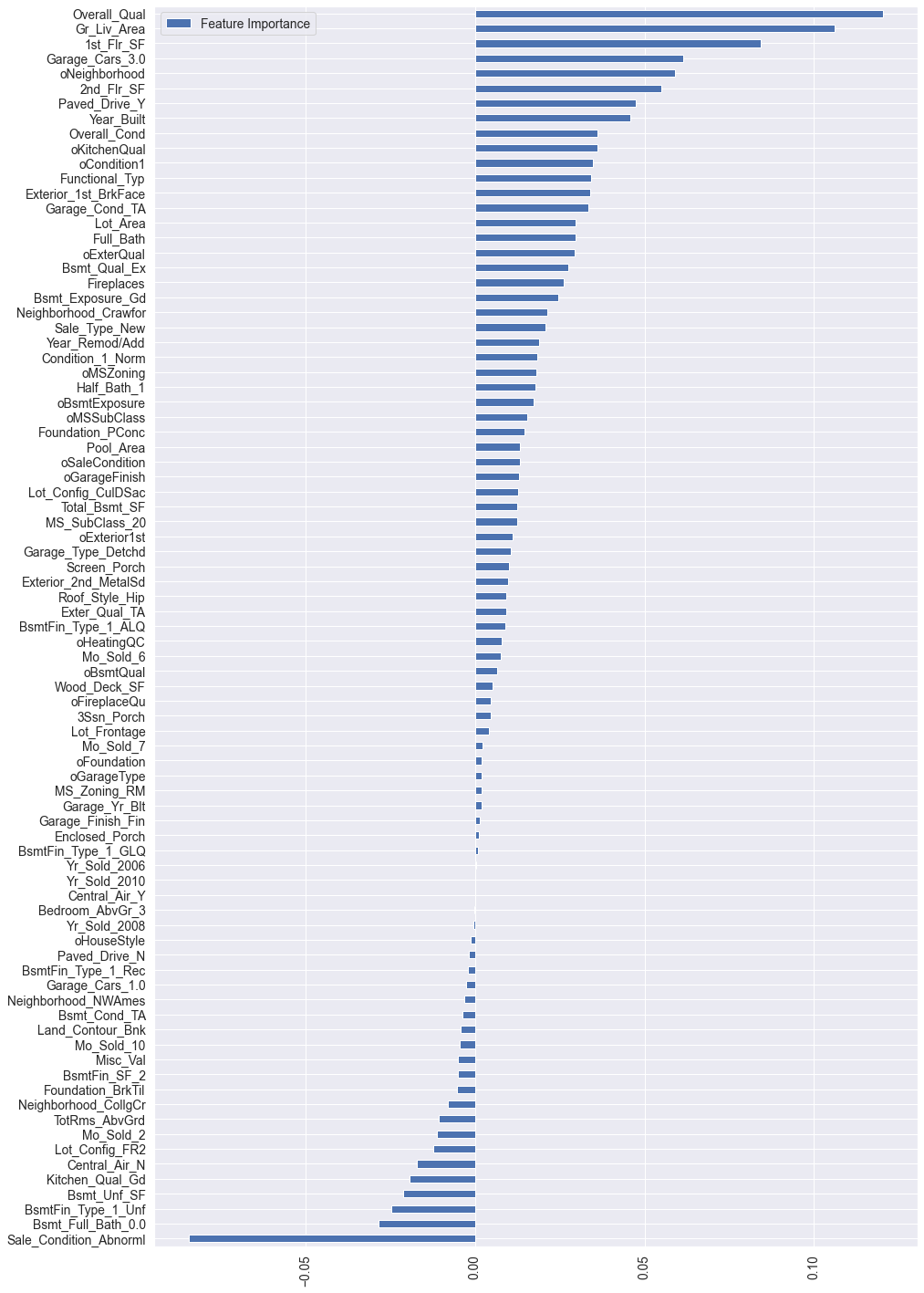
full["oSaleCondition"] = full.Sale\_Condition.map({'AdjLand':1, 'Abnorml':2, 'Alloca':2, 'Family':2, 'Normal':3, 'Partial':4})

I have converted the values of dozens of columns in total, in order to facilitate classification, you can refer to the code for details

**Feature combination**

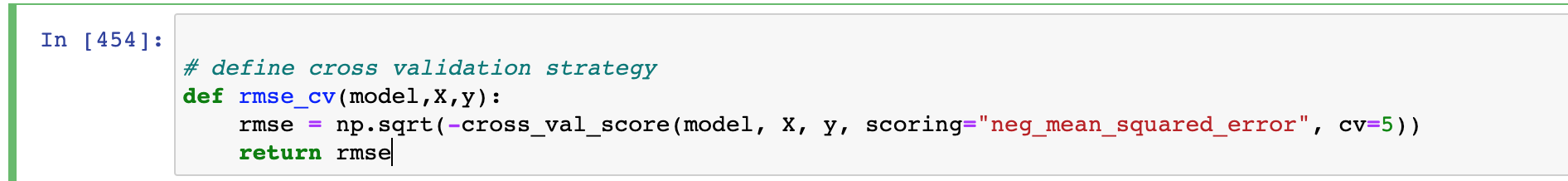
合并原始功能通常会产生意想不到的结果。 但是，此数据集中有许多原始特征，不可能将所有特征一一合并，因此这里我们首先使用Lasso进行特征筛选，然后选择一些更重要的特征进行组合





Finally, these features were added, which also included many other attempts. For details, please refer to the functions and classes related to add\_feature

PCA是非常重要的部分，它可以大大提高最终分数。 因为我添加的这些新功能与原始功能高度相关，所以这可能导致强大的多重共线性（Multicollinearity），而PCA可以修饰这种关联。 因为此处使用PCA的目的不是降低尺寸，所以n\_components使用的尺寸与原始尺寸相似。 这是我多方实验的结果，也就是说，在前面添加了XX个功能，然后缩小为XX个尺寸.

First define the cross-validation evaluation index of RMSE

**model selection**：12 algorithms and 5-fold cross-validation are used to evaluate the baseline effect:

**LinearRegression**

**Ridge**

**Lasso**

**Random Forrest**

**Gradient Boosting Tree**

**Support Vector Regression**

**Linear Support Vector Regression**

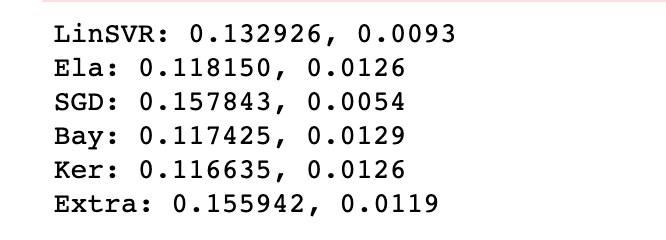
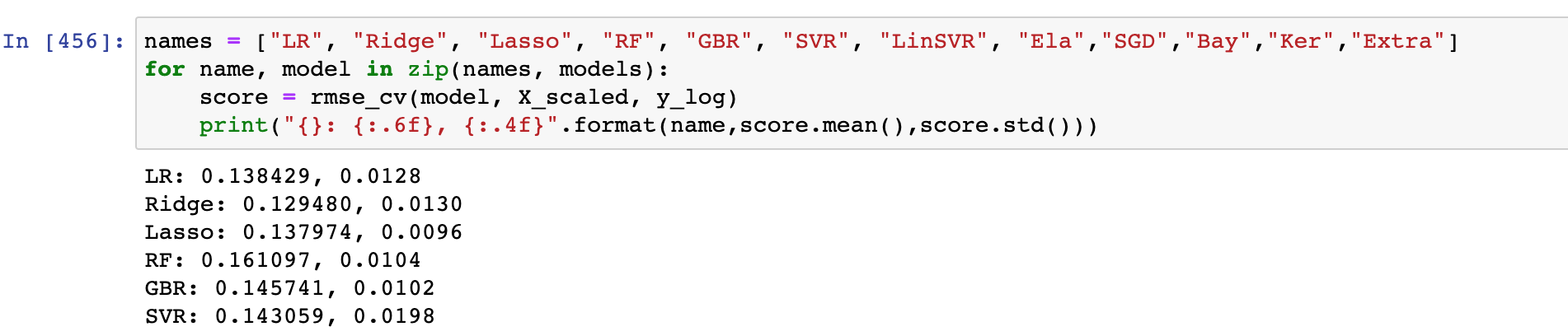
**ElasticNet**

**Stochastic Gradient Descent**

**BayesianRidge**

**KernelRidge**

**ExtraTreesRegressor**



除了测试结果外，我的想法是：树模型通常不如线性模型，这可能是由于get\_dummies /带来的数据稀疏性或数据集不足以使树模型具有最佳性能。

但是，这些模型参数尚未调整。 在做出选择之前，我们应该建立一种调整参数的方法。 我们应始终牢记，评估指标是RMSE.

method to adjust parameter , for all model to use:

grid\_search = GridSearchCV(self.model,param\_grid,cv=5, scoring="neg\_mean\_squared\_error")

grid\_search.fit(X,y)

print(grid\_search.best\_params\_, np.sqrt(-grid\_search.best\_score\_))

grid\_search.cv\_results\_['mean\_test\_score'] = np.sqrt(-grid\_search.cv\_results\_['mean\_test\_score'])

print(pd.DataFrame(grid\_search.cv\_results\_)[['params','mean\_test\_score','std\_test\_score']])

SVR

grid(SVR()).grid\_get(X\_scaled,y\_log,{'C':[11,13,15],'kernel':["rbf"],"gamma":[0.0003,0.0004],"epsilon":[0.008,0.009]})

results stablized at:

std\_test\_score

0 0.001553

1 0.001609

2 0.001555

….

11 0.001617

Kernel

param\_grid={'alpha':[0.2,0.3,0.4], 'kernel':["polynomial"], 'degree':[3],'coef0':[0.8,1]}

grid(KernelRidge()).grid\_get(X\_scaled,y\_log,param\_grid)

results stablized at:

std\_test\_score

0 0.001209

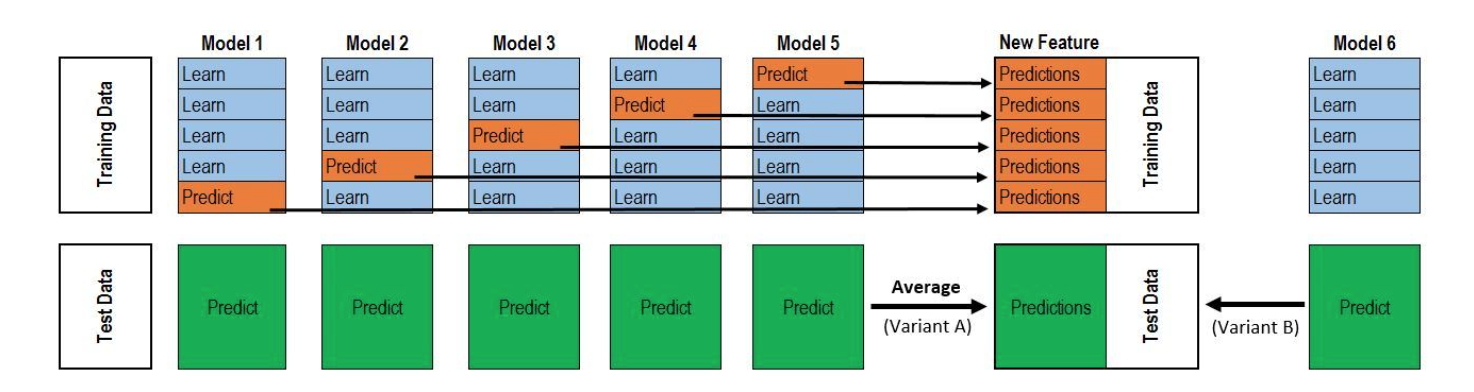
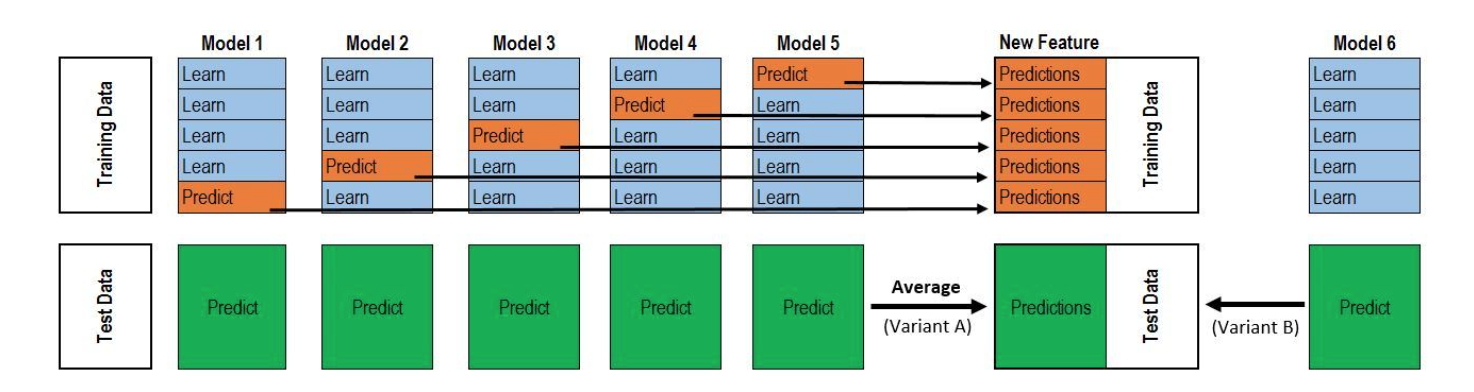
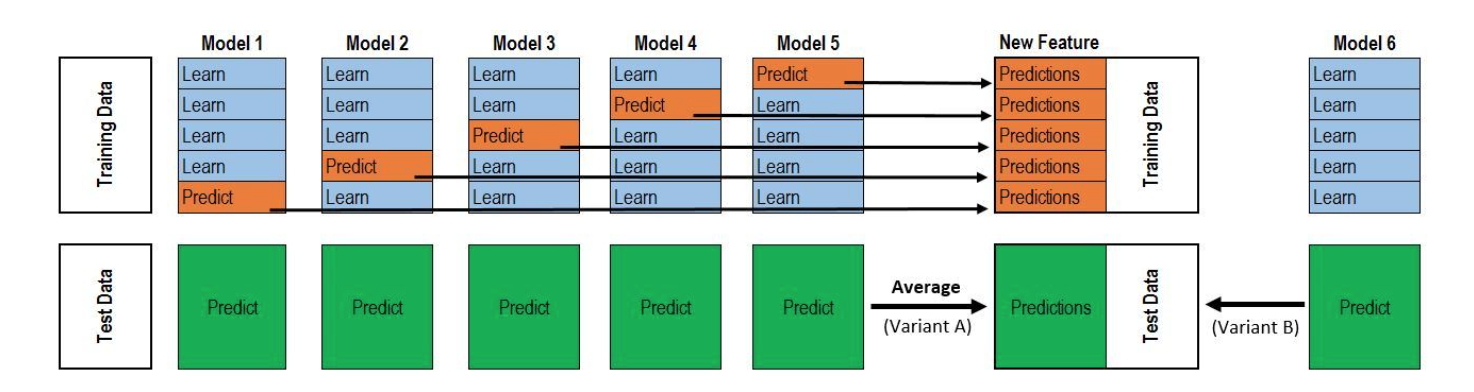
1 0.001243

2 0.001189

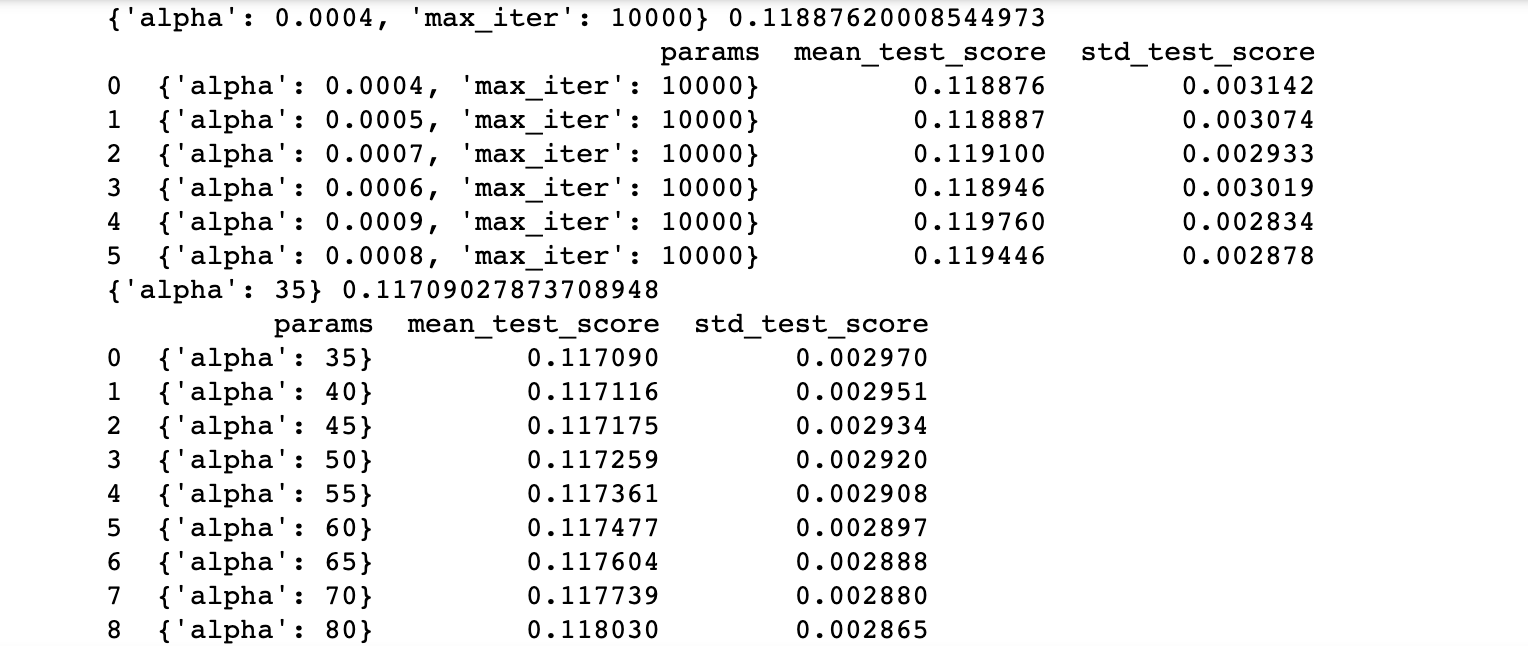
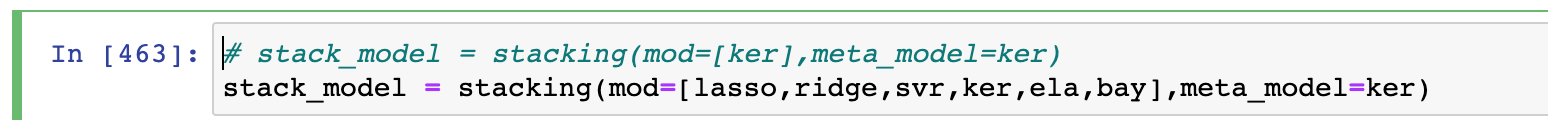
3 0.001210

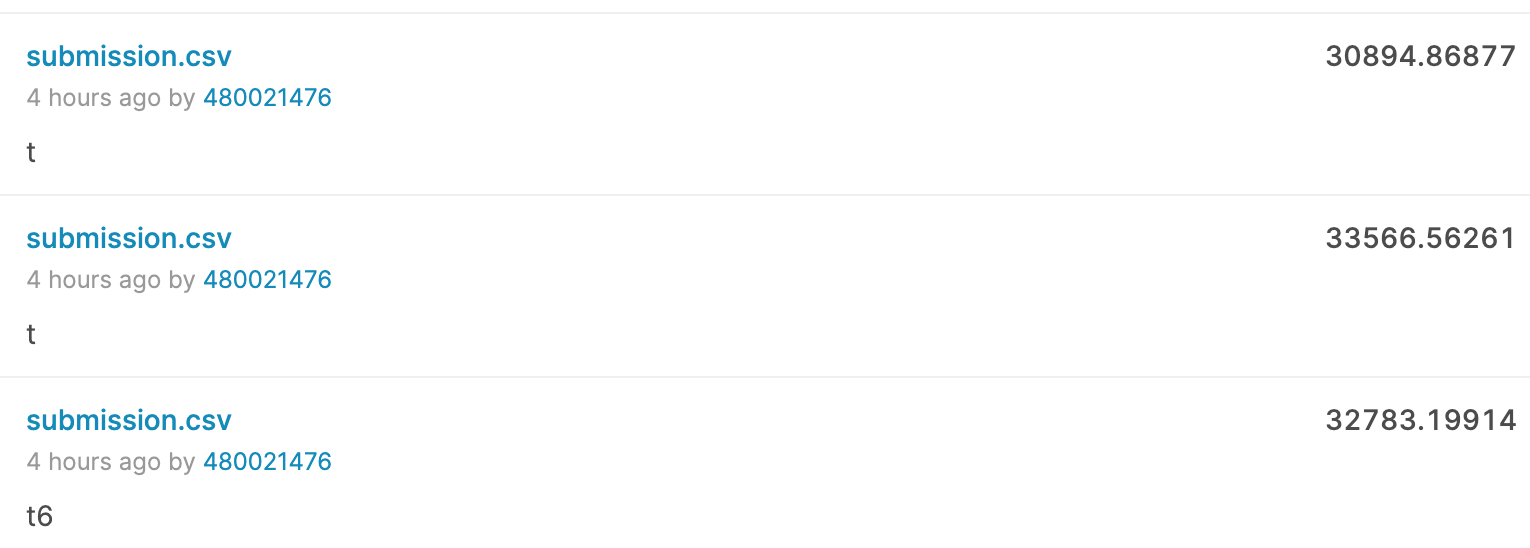
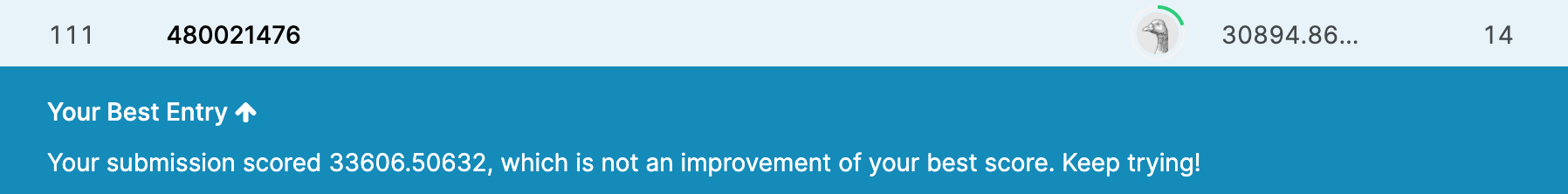
4 0.001181

5 0.001191   
 After many long rounds of testing among all the models, **SVR, KER** are my best 2 models. If I use single model, I will choose from SVR and Kernel Ridge two models.

But I want to combine Weighted average and stacking all 5 models.

In the two-layer stacking as shown in the figure, there are 5 models in the first layer and 1 meta-model in the second layer. The function of the first layer model is to train to obtain a [formula] feature matrix for input to the second layer model training, where n is the number of training data rows and m is the number of the first layer model.

After many long rounds of testing, these six models were finally selected

**Validation set Kaggle results   
**

**Final analysis, conclusion, limitations**

了解数据很重要，清理和转换数据也很重要。 既有离散功能又有连续功能，并且有很多缺失值。 幸运的是，参赛者提供了data\_description.txt文件，该文件描述了每个功能的含义。 了解内容后，可以对大多数缺失值进行平滑插值。

如果我有更多的时间和更多的数据，我想尝试一下神经网络方法。 在训练过程中，只要有足够的输入x和输出y，就可以训练更好的神经网络模型。 在类似的住房价格预测问题中，可以获得更准确的结果。