项目报告

机器学习课程项目：房价预测

2020 年 12 月

摘 要

此次实验，我们小组选择用线性回归模型用于房价的预测。房价预测问题对于大家并不陌生，其特点是多个特征值都对最终价格产生了一定影响。如何处理好众多特征之间的关系、进行特征处理，很是关键。舍弃哪些特征、填充哪些特征值、转换哪些数值、选出相关性哪些更强的特征，关系到最终模型的构建、性能的好坏。利用用手写线性回归以及梯度下降函数进行模型构建与训练。最后用 MSE/RMSE 计算公式对模型评估，初步完成机器学习领域的一个 Toy Project。

关键词：LinearRegression, GradientDescent, House-Price-Prediction

1 介绍

1.1 主要问题

1. 特征值维数众多，信息量很大，需要逐个理解含义。
2. 特征值的分布问题，离散型 or 连续型，同时也-=存在许多缺失的值(NA)。
3. 特征值的有效性问题，不是每个特征值都会对最终的模型起到正向作用，需要手动选择和处理，摸索中前进。
4. 手写线性回归与梯度下降函数，与现成封装好的函数库相比，拟合性能存在上限。

1.2 解决方案

这是一个多特征的回归问题，预测目标是房屋的价格，最终衡量指标是取 log 后的 RMSE。我们要做的就是：选取哪些特征作为模型输入，选择什么模型，使得 RMSE 值尽可能小。

首先从数据观察入手，依次认识并带入实际生活来加深理解。再着手逐步进行数据清洗、drop 掉无用值、离群值 outlier 等，然后进行标签编码，处理数据单位问题。接着进行数据集中训练集测试集的划分、特征重构。再根据手写实现的算法代码进行模型训练，不断调整参数大小与组合、精进模型，最后评估实验性能、考虑不足与收获。

2 方法

2.0 数据观察

0. 数据分析和处理的工作是最考验耐性的，需要一遍遍地去看每列数据代表的意义，让我们更熟悉哪些列可能会影响最终的售价。

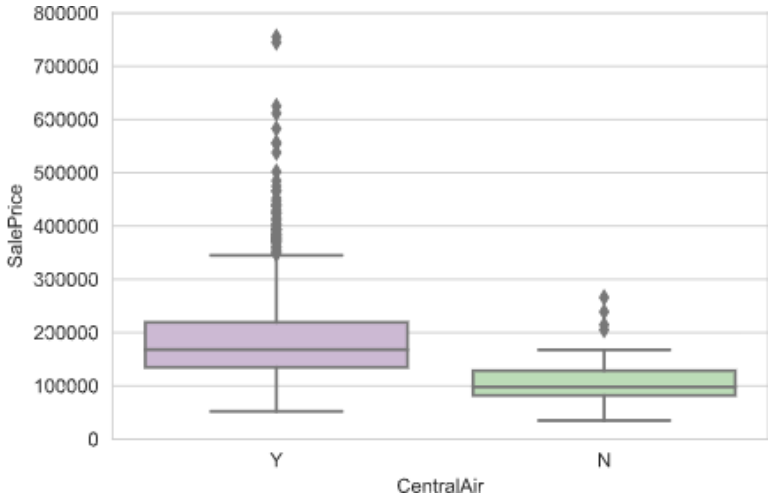
1. 计算 SalePrice 的峰度、偏度，以此了解数据集的分布形态，并借助 seaborn、matplotlib 包进行数据可视化。

2. 分析各项特征与目标变量(SalePrice)的关系

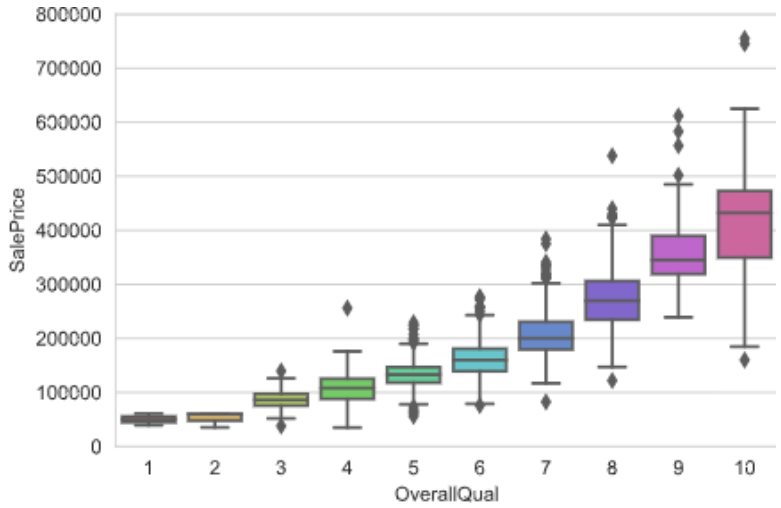
- 代码实现（例）：

```
# 人群中的评价 OverallQual V.S. 售价 SalePrice
data = pd.concat([data_train['SalePrice'], data_train['OverallQual']], axis = 1)
sns.set_style('whitegrid')
fig = sns.boxplot(x = data['OverallQual'], y = data['SalePrice'], data = data, palette = 'hls')
fig.axis(ymin = 0, ymax = 800000)
print(data.groupby(['OverallQual']).describe().T)
# 地上面积 GrLivArea V.S. 售价 SalePrice
data = pd.concat([data_train['SalePrice'], data_train['GrLivArea']], axis = 1)
sns.set_palette('muted')
sns.jointplot(x = 'GrLivArea', y = 'SalePrice', data = data, kind = 'reg', space = 0.5, ratio = 5)
```

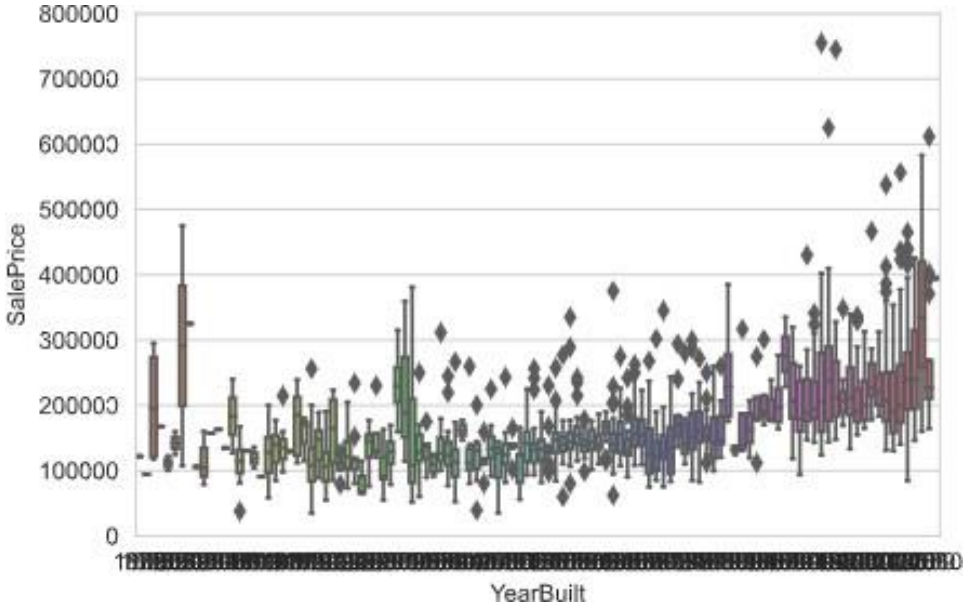
（1）是否有中央空调 CentralAir V.S. 售价 SalePrice



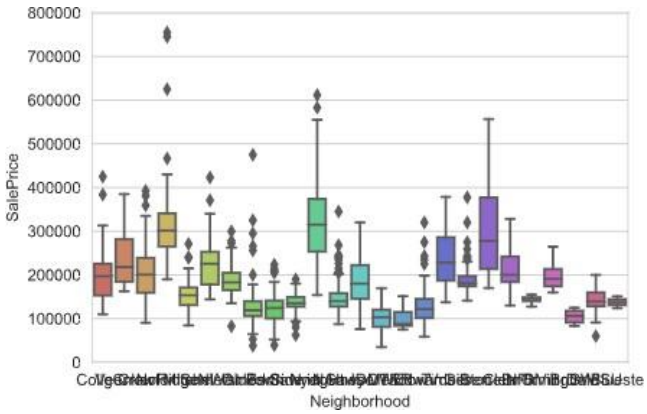
（2） 人群中的评价 OverallQual V.S. 售价 SalePrice



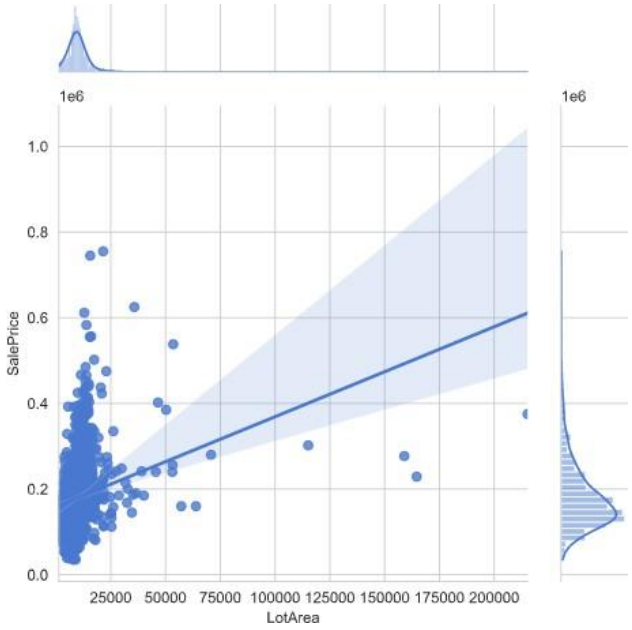
（3） 建造年份 YearBuilt V.S. 售价 SalePrice



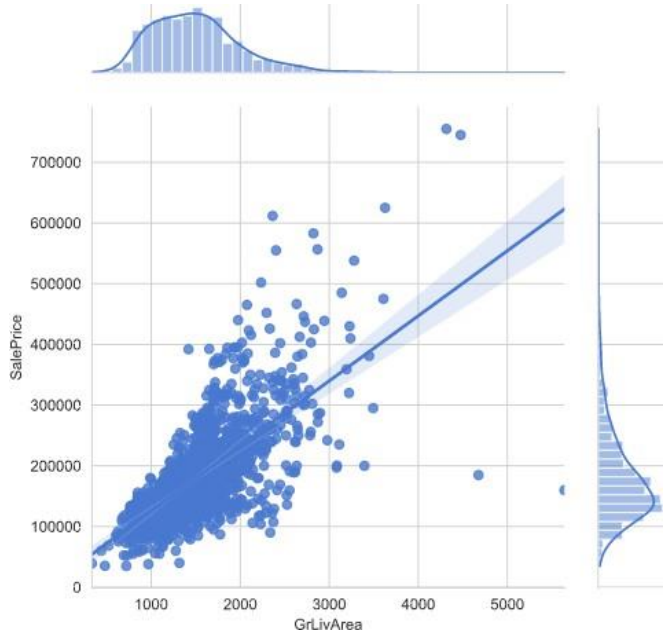
（4） 社区 Neighborhood V.S. 售价 SalePrice



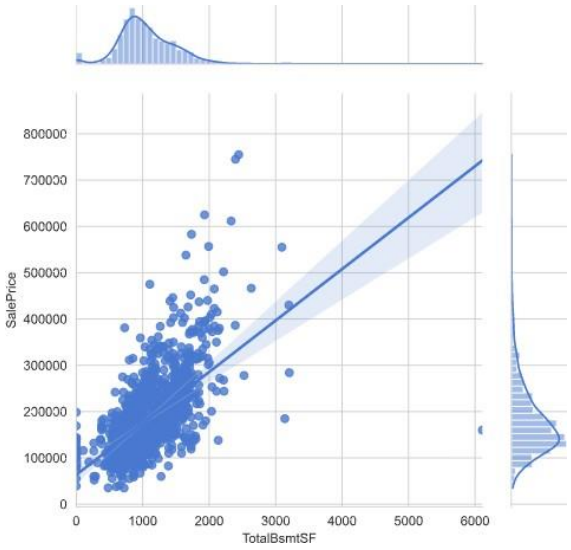
（5） 地表面积 LotArea V.S. 售价 SalePrice



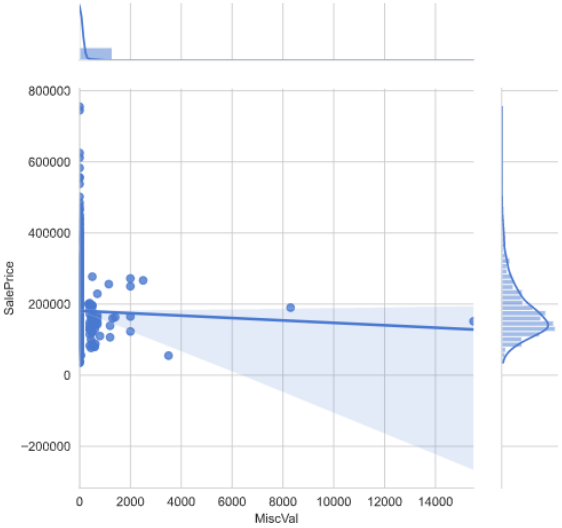
（6）地上面积 GrLivArea V.S. 售价 SalePrice



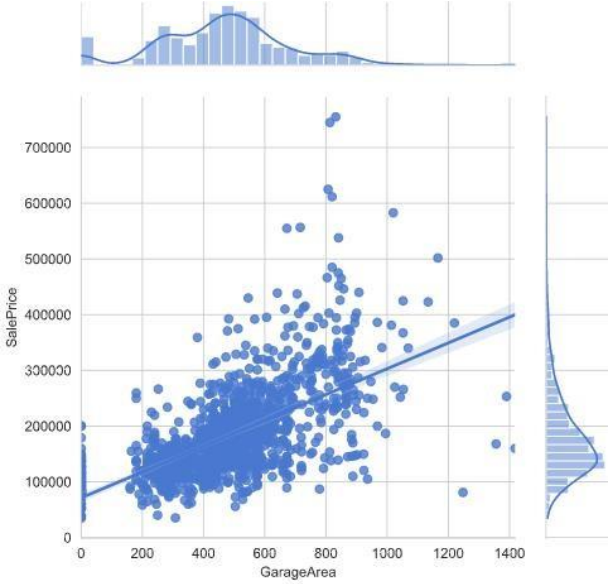
（7）地下室总面积 TotalBsmtSF V.S. 售价 SalePrice



（8） 附加值 MiscVal V.S. 售价 SalePrice



（9） 车库 Garage V.S. 售价 SalePrice



2.1 数据清洗

1. 总共有 1460 条训练数据，除 ID、价格以外，其余列均为特征

2. 价格数据正常，没有为 0 的数据，大都分布在 50000-500000，最后作为预测目标时，需要取对数

3. 计算各列为 NaN 的数据量，如果超过一半则直接去掉这一列

4. drop 掉一部分特征：'Alley', 'PoolQC', 'Fence', 'MiscFeature'等

代码实现（例）：

```
data_train = data_train.drop(['Alley'], axis = 1)
data = data_train.drop(['Id'], axis = 1)
```

5. 观察数据可以看到，大部分缺失的值均表示不存在该类别，如' No basement''No garage'，所以直接用 None 代替。像‘MSZoning’、‘Utilities’类型也选择用众数来填充

代码实现（例）：

```
data['GarageCond'].fillna('None', inplace = True)
data['GarageCond'].value_counts()
```

6. 有的列受其他列的影响，比如'LotFrontage'明显和'Neighborhood'是相关的，所以我们选择取相同'Neighborhood'的'LotFrontage'的中位数来填充

7. 有的列有明显存在意义，不可能为 0。比如‘Electrical’，每套房子肯定都存在电路系统，以及‘Exterior1st’，‘Exterior2nd’，外墙肯定是由某种材料覆盖的，不可能为 0。在这种情况下，我们可以考虑采用众数（即出现次数最多的数）来填充。

2.2 特征重构

0. 数据清洗和重建特征部分参考了 Kaggle 上的一些做法

1. 直观来看，我们的数据主要分为两类，数值型和字符串型，数值型又可能包含用数字表示的类型，比如‘OverallQual’列，用 10 代表 very excellent， 9 代表 excellent，1 代表 ver poor，这是用数字表示的类别类型，不能简单的当作数值类型进行归一化等操作。

2. 对于类别类型，我们通常有两种编码方式：LabelEncoder 和 OneHotEncoder，两者的试用场景不同，如果类别特征没有大小意义（如颜色类包含红、绿、黑），适合用 one-hot 编码，但如果有大小意义（如大小类包含 S、M、L、XL、XXL），选择 LabelEncoder。

3. LabelEncoder 对不连续的数字或者文本进行编号，将 Label 标准化。在训练模型之前，我们通常都要对训练数据进行一定的处理。将类别编号就是一种常用的处理方法， 比如把类别“ 男”，“ 女” 编号为 0 和 1 。可以使用 sklearn.preprocessing 中的 LabelEncoder 处理这个问题。

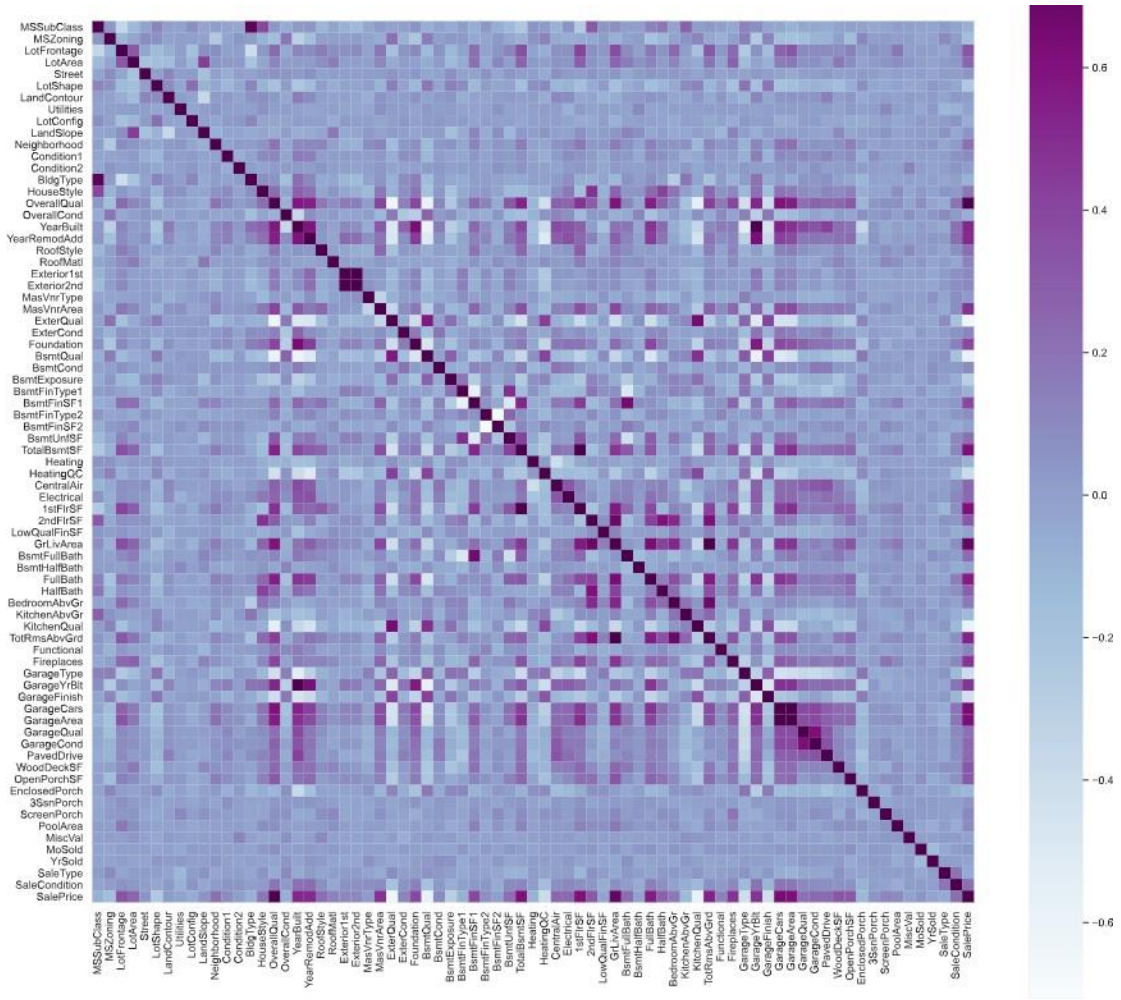
代码实现（例）：

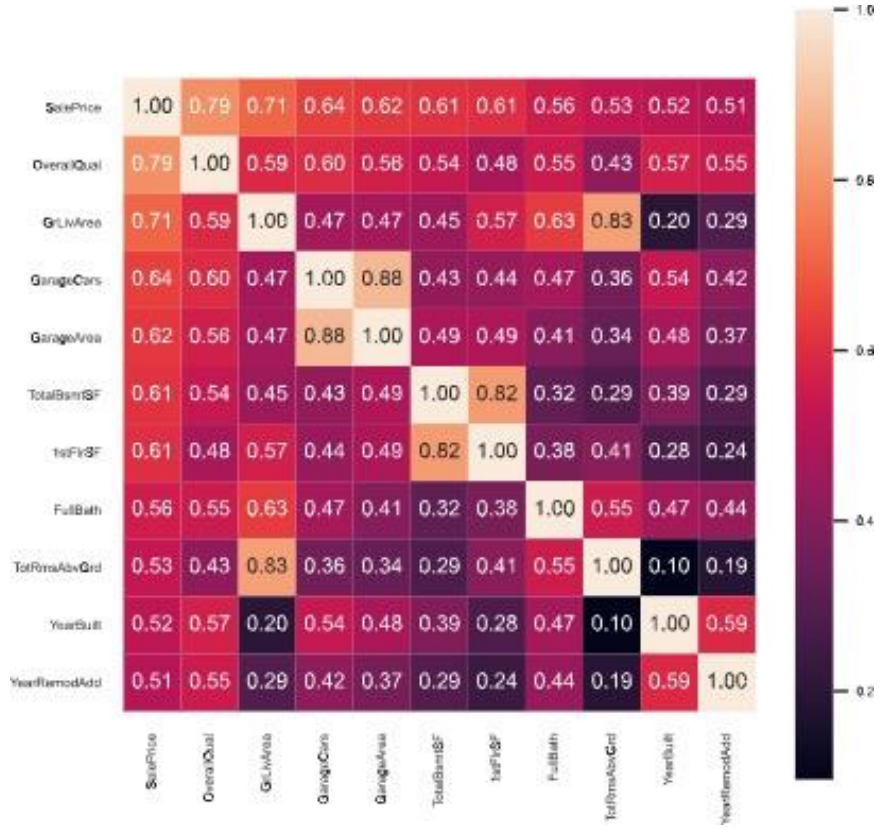
```
from sklearn import preprocessing
for col in data.columns:
    if col in features:
        label = preprocessing.LabelEncoder()
        data[col] = label.fit_transform(data[col])
    else:
        data[col].fillna(data[col].mean(),inplace = True)
```

2.. 画热力图寻求关系，画出各个变量之间的相关性程度热力图-图 1

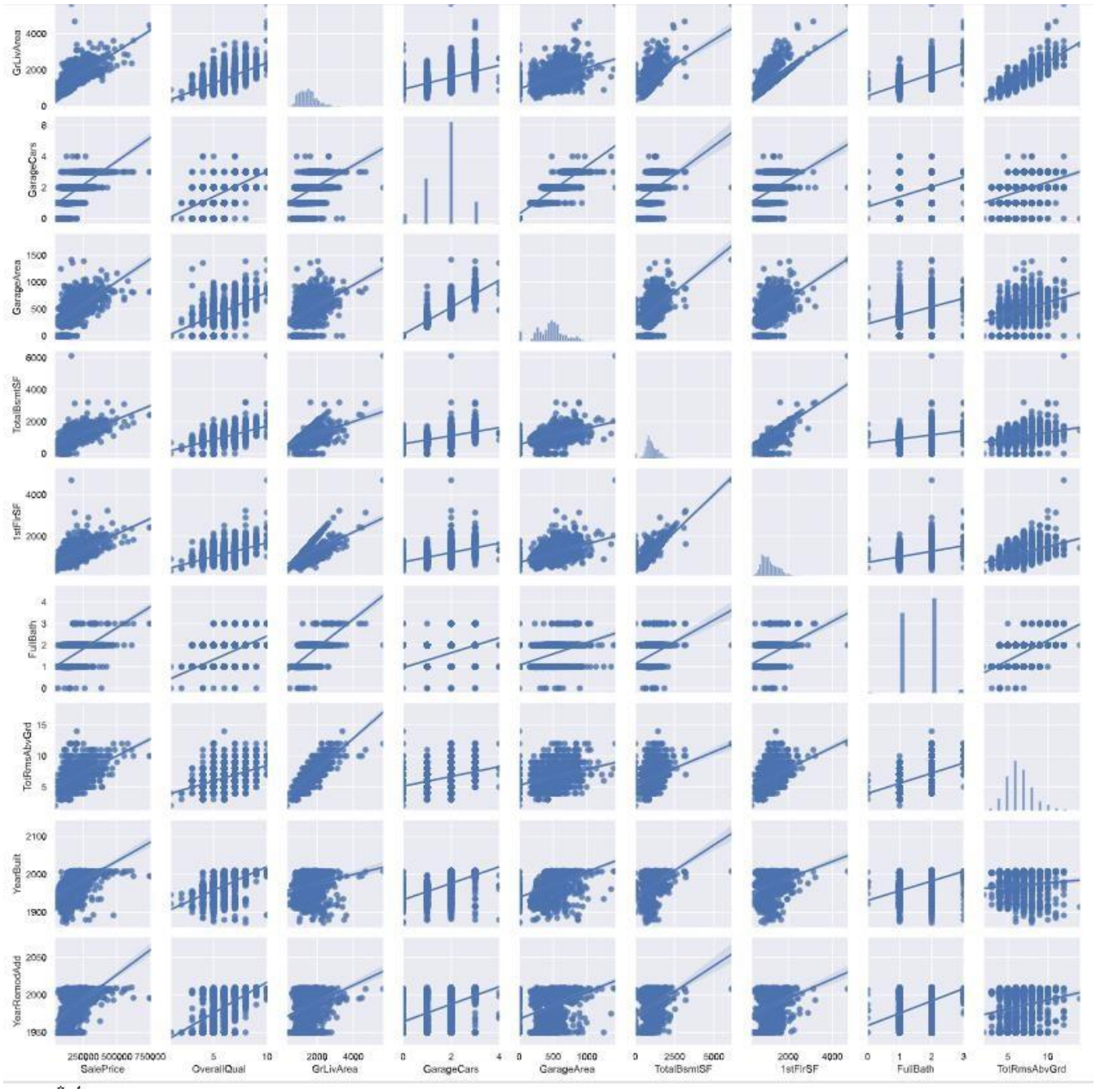
3. 根据热力图，挑出颜色最深（关系最强）的 11 个特征，画关系矩阵显示-图 2

4. 绘画出这 11 个变量 关系点图—图 3


MSSubClass
MSZoning
LotFrontage
LotArea
Street
LotShape
LandContour
Utilities
LotConfig
LandSlope
Neighborhood
Condition1
Condition2
BldgType
HouseStyle
OverallQual
OverallCond
YearBuilt
YearRemodAdd
RoofStyle
RoofMatl
Exterior1st
Exterior2nd
MasVnrType
MasVnrArea
ExterQual
ExterCond
Foundation
BsmtQual
BsmtCond
BsmtExposure
BsmtFinType1
BsmtFinSF1
BsmtFinType2
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
Heating
HeatingQC
CentralAir
Electrical
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
KitchenQual
TotRmsAbvGrd
Functional
Fireplaces
GarageType
GarageYrBlt
GarageFinish
GarageCars
GarageArea
GarageQual
GarageCond
PavedDrive
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SaleType
SaleCondition
SalePrice
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


SalePrice 1.00 0.79 0.71 0.64 0.62 0.61 0.61 0.56 0.53 0.52 0.51
OverallQual 0.79 1.00 0.59 0.60 0.56 0.54 0.48 0.55 0.43 0.57 0.55
GrLivArea 0.71 0.59 1.00 0.47 0.47 0.45 0.57 0.63 0.83 0.20 0.29
GarageCars 0.64 0.60 0.47 1.00 0.88 0.43 0.44 0.47 0.36 0.54 0.42
GarageArea 0.62 0.56 0.47 0.88 1.00 0.49 0.49 0.41 0.34 0.48 0.37
TotalBsmtSF 0.61 0.54 0.45 0.43 0.49 1.00 0.82 0.32 0.29 0.39 0.29
1stFlrSF 0.61 0.48 0.57 0.44 0.49 0.82 1.00 0.38 0.41 0.28 0.24
FullBath 0.56 0.55 0.63 0.47 0.41 0.32 0.38 1.00 0.55 0.47 0.44
TotRmsAbvGrd 0.53 0.43 0.83 0.36 0.34 0.29 0.41 0.55 1.00 0.10 0.19
YearBuilt 0.52 0.57 0.20 0.54 0.48 0.39 0.28 0.47 0.10 1.00 0.59
YearRemodAdd 0.51 0.55 0.29 0.42 0.37 0.29 0.24 0.44 0.19 0.59 1.00
SalePrice
OverallQual
GrLivArea
GarageCars
GarageArea
TotalBsmtSF
1stFlrSF
FullBath
TotRmsAbvGrd
YearBuilt
YearRemodAdd
1.0
0.8
0.6
0.4
0.2

```
sns.set()
# 将 kind 参数设置为 reg 会为非对角线上的散点图拟合出一条回归直线，更直观地显示变量之间的关系
sns.pairplot(data = data[features], kind = 'reg',size=2)
plt.show()
```



2.3 线性回归算法实现

本次项目限制要使用线性回归的方法进行模型构建，代价函数和梯度函数的代码细节需自己编写完成。

```
# 模型定义
class linear_regression():
    def fitness(self, Date_X_input, Date_Y, learning_rate=0.3, lamda=0.003):
        # 获取样本个数、样本的属性个数
        sample_num, property_num = Date_X_input.shape
        Date_X = np.c_[Date_X_input, np.ones(sample_num)]

        # 初始化待调参数 theta
        self.theta = np.zeros([property_num + 1, 1])
        # 最多迭代次数
        Max_count = int(1e8)
        # 上一次得到较好学习误差的迭代学习次数
        last_better = 0
        # 上一次得到较好学习误差的误差函数值
        last_Jerr = int(1e8)
        # 定义在得到较好学习误差之后截止学习的阈值
        threshold_value = 1e-8
        # 定义在得到较好学习误差之后截止学习之前的学习次数
        threshold_count = 10
        for step in range(0, Max_count):
            # 预测值
            predict = Date_X.dot(self.theta)
            # 损失函数
            J_theta = sum((predict - Date_Y) ** 2) / (2 * sample_num)
            self.theta -= learning_rate * (lamda * self.theta + (Date_X.T.dot(predict - Date
_Y)) / sample_num)          # 更新参数 theta
            # 检测损失函数的变化值，提前结束迭代
            if J_theta < last_Jerr - threshold_value:
                last_Jerr = J_theta
```

```
            last_better = step
        elif step - last_better > threshold_count:
            break
        # 定期打印，方便用户观察变化
        if step % 100 == 0:
            print("迭代次数 %s 代价 J %.8f" % (step, J_theta))
    def predicted(self, X_input):
        sample_num = X_input.shape[0]
        X = np.c_[X_input, np.ones(sample_num, )]
        predict = X.dot(self.theta)
        return predict
# 对数据集中的样本属性进行分割，制作 X 和 Y 矩阵
def property_label(pd_data):
    # 行数、列数
    row_num = pd_data.shape[0]
    column_num = len(pd_data.iloc[0, 0].split())
    X = np.empty([row_num, column_num - 1])
    Y = np.empty([row_num, 1])
    for i in range(0, row_num):
        row_array = pd_data.iloc[i, 0].split()
        X[i] = np.array(row_array[0:-1])
        Y[i] = np.array(row_array[-1])
    return X, Y
# 把特征数据进行标准化为均匀分布
def  standardization(X_input):
    Maxx = X_input.max(axis=0)
    Minx = X_input.min(axis=0)
    X = (X_input - Minx) / (Maxx - Minx)
return X, Maxx, Minx
```

2.4. 模型训练与调参

通过不断调整学习率 learning rate、正则化参数 lambda 这两个变量得到最优解

3 实验性能

1. 在测试集上的表现

横坐标是样本标签，纵坐标是预测值与真实值，曲线越贴合，说明性能越好。

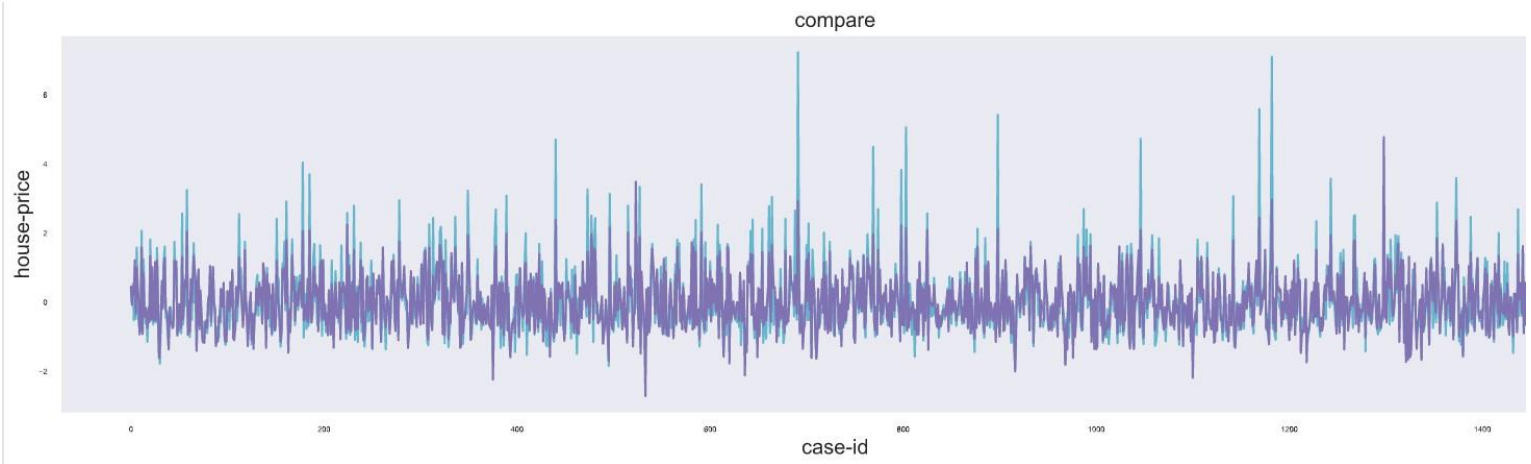


图 1 预测-真实对照图

代码实现：

```
plt.style.use({'figure.figsize' : (20, 5)})
plt.figure(facecolor='w')
plt.plot(y_scaled, 'c-', lw=1.6, label=u'real')
plt.plot(Date_predict, 'm-', lw=1.6, label=u'predict')
plt.legend(loc='best')
plt.title(u'compare', fontsize=15)
plt.xlabel(u'case-id', fontsize=15)
plt.ylabel(u'house-price', fontsize=15)
plt.grid()
plt.show()
```

2. 评估函数

在数据处理的时候将 saleprice 做了变化，原来 saleprice 的数据分布基本在十万级的区间，变换成了十几的区间，需要先将预测的 saleprice 执行逆变换，再与数据集中的真实的 saleprice 比较，计算两者之间的均方差

代码实现：

```
# 定义评估函数
# 均方差：MSE 的值越小，预测模型具有更好的精确度
def MSE(y, y_pre):
```

```
    return np.mean((y - y_pre) ** 2)

# 均方根误差：RMSE 的值越小，预测模型具有更好的精确度
def RMSE(y, y_pre):
    return np.sqrt(((y - y_pre) ** 2).mean())
# print("MSE:")
# print(MSE(y_scaled,Date_predict))
print("RMSE:")
print(RMSE(y_scaled,Date_predict))
```

```
✓ 0.3s
RMSE:
0.22888367121378667
```

均方根误差 RMSE 的值越小，预测模型具有更好的精确度。

4 讨论(结论)

通过合作完成机器学习的项目作业，在实践中巩固了课上所学，很有意义。机器学习是当下计算机科学领域一大热门方向，本学期可以选修这么一门贴合前沿的专业课，非常幸运。

具体到本项目，应用线性回归模型来预测房价问题。借助本项目，大致了解了研究的流程与基本方法，为今后的深入研究打下基础。

采用回归分析，就是通过纳入多个自变量，达到控制混杂因素的作用。线性回归获得的系数代表的是相关关系，而非因果关联。回归的使用仅能说明数据之前存在关联，但这种关联是否真正代表了两者的内在联系还需要更深入的研究。但是我们无法合理评估所有可能的因素，从而导致回归的结果不准确。

本项目有八十多个特征值，如何取舍与重构特征，以便更好地训练模型，是一门学问，非常有必要花费时间研究。

相比于其他主流的模型选择，比如随机森林、岭回归、罗森回归，小组选择了手写线性回归模型，虽然训练效果有很大提升空间，但也能从参数不断调优的过程中学到很多。如果还有时间，用现有的模型框架做一次来与本次线程对比，也是很有意义的。

关于线性回归模型的评估，我们小组查找了许多不同的评判体系，通过比较不同衡量指标的侧重点与原理，更加等深入地理解了模型训练与应用。对于简单

线性回归，目标是找到 a,b　使得下式尽可能小：

$$\sum_{i=1}^{m}(y^{(i)}-ax^{(i)}-b)^2$$

但问题是这个衡量标准和 m 相关，比如当 10000 个样本误差累积是 100，而 1000 个样本误差累积却达到了 80，虽然 80 小于 100，但我们却不能说第二个模型优于第一个。

改进 1： 对式子除以 m，使得其与测试样本 m 无关。就得到了均方误差 MSE，但又有一个问题，之前算这个公式时为了保证其每项为正，且可导，我们对式子加了一个平方。但这可能会导致量纲的问题，如房子价格为万元，平方后就成了万元的平方。

改进 2： 对 MSE 开方，使量纲相同，就得到了均方根误差 RMSE，MSE 与 RMSE 的区别仅在于对量纲是否敏感。

以上就是此次项目作业的流程与心得，不得不说，有汗水也有满满的收获。感谢金老师每次课的倾心传授，尤其是晚上的实验课，老师辛苦了。也感谢各位助教学长，不厌其烦地解答了作为小白的许多问题，给了项目很多有用的意见。没有老师、学长双力相助，不可能有成果的产出，再次感恩！

5 代码运行请看项目根目录下的 README 文件