实验一：北京二手房价格预测

学号 ：2018329621262　　姓名：刘恒玮

一、实验目的

理解线性回归模型，使用线性回归模型解决实际问题。

二、实验环境

PC 机，Python，pandas 库，sklearn 库

三、实验内容

给定训练数据，利用线性回归模型预测北京二手房价格。

四、实验原理（算法、公式推导）

线性回归（Linear Regression）是一种通过属性的线性组合来进行预测的线性模型，其目的是找到一条直线或者一个平面或者更高维的超平面，使得预测值与真实值之间的误差最小化。给定数据集 $D=\{(x_1,y_1),(x_2,y_2),...,(x_m,y_m)\}$，其中 $x_i=(x_{i1};x_{i2};...;x_{id}), y_i\in R$。m 是样本数，d 是属性维度。线性回归试图学得：$f(x_i)=\omega^T x_i+b$，使得 $f(x)\approx y_i$。预测值和真实值之间都肯定存在差异，对于每个样本：$y_i=\omega^T x_i+\varepsilon_i$，假设误差是独立同分布的，并且服从于高斯分布，则预测值的条件概率为：$p(y_i\mid x_i;\omega)=\frac{1}{\sqrt{2\pi\sigma}}\exp(-\frac{(y_i-\omega^T x_i)^2}{2\sigma^2})$，最终可以得到代价函数 $J(\omega)=\frac{1}{2}\sum_{i=1}^{m}(y_i-\omega^T x_i)^2$，这个代价函数也称为平方误差代价函数，我们的目标是使这个代价函数越小越好。

五、实验步骤（分析过程）

1．导入相关的库，并利用 pandas 库读取训练集和测试集的数据，并设置编码方式为 iso8859-1。

[1]:
```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import itertools
df_train = pd.read_csv("beijing.csv", encoding='iso-8859-1', low_memory = False)
df_test = pd.read_csv("test.csv", encoding='iso-8859-1', low_memory = False)
```

2. 删除训练集中不相关的数据列 URL 和 ID，以及缺失过多的数据列 DOM。

[2]:
```
df_train.drop(['url','id'],axis=1,inplace = True)#删除URL和id两列
df_train.drop(['DOM'],axis=1,inplace = True) #缺失值过多 直接删去该列
```

3. 查看各特征值的数据类型以及数据缺失情况。

[3]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255081 entries, 0 to 255080
Data columns (total 23 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   Lng                  255081 non-null  float64
 1   Lat                  255081 non-null  float64
 2   Cid                  255081 non-null  int64
 3   tradeTime            255081 non-null  object
 4   followers            255081 non-null  int64
 5   totalPrice           255081 non-null  float64
 6   price                255081 non-null  int64
 7   square               255081 non-null  float64
 8   livingRoom           255081 non-null  object
 9   drawingRoom          255081 non-null  object
 10  kitchen              255081 non-null  int64
 11  bathRoom             255081 non-null  object
 12  floor                255081 non-null  object
 13  buildingType         253445 non-null  float64
 14  constructionTime     255081 non-null  object
 15  renovationCondition  255081 non-null  int64
 16  buildingStructure    255081 non-null  int64
 17  ladderRatio          255081 non-null  float64
 18  elevator             255061 non-null  float64
 19  fiveYearsProperty    255061 non-null  float64
 20  subway               255061 non-null  float64
 21  district             255081 non-null  int64
 22  communityAverage     254703 non-null  float64
dtypes: float64(10), int64(7), object(6)
memory usage: 44.8+ MB
```

可以看出总共有 0-22 总共 23 个特征值，其中 13buildingType、18elevator、19fiveYearsProperty、20subway、22community 总共五个特征值数据有缺失，其中 8livingRoom、9drawingRoom、11bathRoom、12floor、14constructionTime 总共五个特征值的数据类型为 object，要进行数据类型转化。

4. 对五个数据有缺失的特征值进行处理，删去 elevator、subway、buildingType、fiveYearsProperty 这四列有缺失的行，用平均值填充 community 特征值缺失的单元格。

```
[4]: df_train.dropna(subset=['elevator','subway','buildingType','fiveYearsProperty'], axis = 0, inplace = True)
     df_train['communityAverage'].fillna(df_train['communityAverage'].mean(), inplace=True)
```

5. 将 tradeTime 列中的年、月、日信息提取出来，重新生成 year 和模拟退火、两列，并且只取 2012-2017 年的数据，删去其他不具代表性的少量数据，处理完成后将 tradeTime 列删去。

```
[5]: df_train['tradeTime'] = pd.to_datetime(df_train['tradeTime'])
     df_train['year'] = df_train['tradeTime'].dt.year#将交易时间的数据类型转为int
     df_train['month'] = df_train['tradeTime'].dt.month#将交易时间的数据类型转为int
     df_train.drop(df_train[df_train['year']  < 2012].index, inplace = True)
     df_train.drop(df_train[df_train['year']  > 2017].index, inplace = True)#只取2012-2017年的数据
     df_train.drop(['tradeTime'],axis=1,inplace = True)
```

6. Floor 和 constructionTime 列中的数据存在乱码，用正则表达式提取数字，并将数据类型转化为 int，若有空缺值，则用最多的数值填充。

```
[7]: df_train['floor'] = df_train['floor'].str.extract(r'([0-9]+)')
     df_train['floor'] = df_train['floor'].astype('int64')
     df_train['constructionTime'] = df_train['constructionTime'].str.extract(r'([0-9]+)')
     df_train['constructionTime'].fillna(2004, inplace=True)#用最多的值填充
     df_train['constructionTime'] = df_train['constructionTime'].astype('int64')
```

7. 将 8livingRoom、9drawingRoom、11bathRoom 三列的数据类型转为 int。

```
[8]: df_train['livingRoom'] = df_train['livingRoom'].astype('int64')
     df_train['drawingRoom'] = df_train['drawingRoom'].astype('int64')
     df_train['bathRoom'] = df_train['bathRoom'].astype('int64')
```

8. 利用 pandas 中的 describe()函数查看各特征值的统计信息，并对 totalPrice 和 Price 只取 5%-95%的数据 删去其他不具代表性的少量数据，最后用均值填充整

张表的空缺值。

[9]:
```
df_train.describe(percentiles=[.05,.5,.95])
```

[9]:

	Lng	Lat	Cid	followers	totalPrice	price	square	livingRoom	drawingRoom	kitchen	...
count	248316.000000	248316.000000	2.483160e+05	248316.000000	248316.000000	248316.000000	248316.000000	248316.000000	248316.000000	248316.000000	...
mean	116.419135	39.949551	1.124621e+12	16.990307	351.350663	44016.620451	82.781325	2.003483	1.168410	0.995316	...
std	0.111336	0.091121	1.257757e+12	33.962348	225.123110	21655.712854	35.994052	0.767428	0.518386	0.100249	...
min	116.072514	39.627030	1.111027e+12	0.000000	0.100000	1.000000	7.370000	0.000000	0.000000	0.000000	...
5%	116.235582	39.801739	1.111027e+12	0.000000	129.000000	18628.000000	42.000000	1.000000	0.000000	1.000000	...
50%	116.416894	39.934530	1.111027e+12	5.000000	296.000000	39167.000000	74.110000	2.000000	1.000000	1.000000	...
95%	116.643924	40.094293	1.111027e+12	72.000000	750.000000	87049.000000	148.990000	3.000000	2.000000	1.000000	...
max	116.711337	40.252758	1.184867e+14	1143.000000	4900.000000	156250.000000	922.700000	8.000000	5.000000	3.000000	...

8 rows × 24 columns

[10]:
```
df_train.drop(df_train[df_train['totalPrice']  < 129].index, inplace = True)
df_train.drop(df_train[df_train['totalPrice']  > 750].index, inplace = True)#只取5%-95%的数据 删去其他不具代表性的少量数据
df_train.drop(df_train[df_train['square']  < 42].index, inplace = True)
df_train.drop(df_train[df_train['square']  > 149].index, inplace = True)#只取5%-95%的数据 删去其他不具代表性的少量数据
```

[11]:
```
df_train.fillna(df_train.mean(), inplace=True)
```

9. 用上述相同的流程处理测试集的数据。

[14]:
```
df_test.drop(['url','id','DOM'],axis=1,inplace = True)#删除URL和id两列
df_test['tradeTime'] = pd.to_datetime(df_test['tradeTime'])
df_test['year'] = df_test['tradeTime'].dt.year#将交易时间的数据类型转为int
df_test['month'] = df_test['tradeTime'].dt.month#将交易时间的数据类型转为int
df_test.drop(['tradeTime'],axis=1,inplace = True)
df_test['livingRoom'] = df_test['livingRoom'].str.extract(r'([0-9]+)')
df_test['livingRoom'].fillna(2, inplace=True)#用最多的值填充
df_test['livingRoom'] = df_test['livingRoom'].astype('int64')
df_test['drawingRoom'] = df_test['drawingRoom'].str.extract(r'([0-9]+)')
df_test['drawingRoom'].fillna(1, inplace=True)#用最多的值填充
df_test['drawingRoom'] = df_test['drawingRoom'].astype('int64')
df_test['floor'] = df_test['floor'].str.extract(r'([0-9]+)')
df_test['floor'] = df_test['floor'].astype('int64')
df_test['constructionTime'] = df_test['constructionTime'].str.extract(r'([0-9]+)')
df_test['constructionTime'].fillna(2004, inplace=True)#用最多的值填充
df_test['constructionTime'] = df_test['constructionTime'].astype('int64')
df_test.fillna(df_test.mean(), inplace=True)
```

10. 使用 train_test_split 将数据分为训练数据和测试数据。

[16]:
```
from sklearn.model_selection import train_test_split,cross_val_score
X = df_train.drop('totalPrice', axis = 1)
y = df_train['totalPrice']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)
```

11. 对建立好的模型进行交叉验证十次，得到最大的分数。

[17]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X,y)
print("十次交叉验证得到的最大分数score =",max(cross_val_score(lin_reg, X, y, cv=10)))
```

```
十次交叉验证得到的最大分数score = 0.9490913235430991
```

12. 计算训练集中以下各项指标：

MAE（平均绝对误差）表示原始值和预测值之间的差异，它的值是数据集绝对差的均值。

MSE（均方误差）表示原始值和预测值之间的差值，它的值是数据集平均差的平方。

RMSE（均方根误差）为均方根误差。

R-squared（决定系数）表示与原始值相比数值拟合程度的系数。值在 0 到 1 之间，意为百分比。值越高，模型越好。

[18]:
```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
pred = lin_reg.predict(X_test)
mae = metrics.mean_absolute_error(y_test, pred)
mse = metrics.mean_squared_error(y_test, pred)
rmse = np.sqrt(metrics.mean_squared_error(y_test, pred))
r2_square = metrics.r2_score(y_test, pred)
print("MAE = ",mae)
print("MSE = ",mse)
print("RMSE = ",rmse)
print("R2 = ",r2_square)
```

```
MAE =  26.8132389650104
MSE =  1435.5276297852033
RMSE =  37.88835744374785
R2 =  0.925751360501993
```

13. 计算训练集中以下各项指标：

MAE（平均绝对误差）表示原始值和预测值之间的差异，它的值是数据集绝对差的均值。

MSE（均方误差）表示原始值和预测值之间的差值，它的值是数据集平均差的平方。

RMSE（均方根误差）为均方根误差。

R-squared（决定系数）表示与原始值相比数值拟合程度的系数。值在 0 到 1 之间，意为百分比。值越高，模型越好。

[19]:
```
from sklearn import metrics
result = pd.DataFrame()
result =  pd.read_csv('result.txt',sep=" ")
result['pred'] = pd.DataFrame(lin_reg.predict(df_test))
result.dropna(subset=['totalPrice'], axis = 0, inplace = True)#删去有缺失值的行
result['totalPrice'] = result['totalPrice'].astype('float')
result['pred'] = result['pred'].astype('float')
mae = metrics.mean_absolute_error(result['totalPrice'], result['pred'])
mse = metrics.mean_squared_error(result['totalPrice'], result['pred'])
rmse = np.sqrt(metrics.mean_squared_error(result['totalPrice'], result['pred']))
r2_square = metrics.r2_score(result['totalPrice'], result['pred'])
print("MAE = ",mae)
print("MSE = ",mse)
print("RMSE = ",rmse)
print("R2 = ",r2_square)
```

```
MAE =  54.598910478573565
MSE =  409652.12147311325
RMSE =  640.0407186055535
R2 =  -6.515083876103187
```

14. 将预测结果保存为 result.csv。

[20]:
```
df = pd.read_csv("test.csv", encoding='iso-8859-1', low_memory = False)
df['totalPrice'] = pd.DataFrame(lin_reg.predict(df_test))
df.to_csv('result.csv')
```

六．实验结果（训练集 MSE，验证集 MSE 等指标）

训练集各项指标：

```
MAE =  26.8132389650104
MSE =  1435.5276297852033
RMSE =  37.88835744374785
R2 =  0.925751360501993
```

验证集各项指标：

```
MAE =  54.598910478573565
MSE =  409652.12147311325
RMSE =  640.0407186055535
R2 =  -6.515083876103187
```

可以看出训练集中各项指标的值都比较理想，R2 的值更是高达 0.92，可以反映出这是一个较为理想的线性回归模型，而对于验证集则 MAE、MSE、RMSE 等指标较为可以，但是 R2 的数值却相当不理想，但是实际上这个模型的预测效果还是较为准确的。

七．实验心得（碰到的问题，如何解决，你还想到线性回归可以解决什么问题）

通过本次实验，对线性回归的原理有了进一步的了解，并学会了如何建立一个线性回归模型解决一个实际问题。实验过程中，碰到一些训练集数据杂乱并且庞大的问题，首先选择剔除无关的特征值，对特征值的缺失采用平均值填充、剔除等方法处理，并对一些重要的特征选取具有较强代表性的数据，将一些不具代表性的数据剔除，最后要注意的是要将整个数据集中的数据类型转化为数字型。建立好模型后可以通过交叉验证、求 MAE（平均绝对误差）、MSE（均方误差）、RMSE（均方根误差）、R-squared（决定系数）等指标来判断建立的模型的预测效果是否理想。线性回归模型还可以解决录取大学生预测、股票价格走势预测、预测各种品质珠宝玉石价格等问题。