

0. 知识准备

1.导入包集

3. 数据预处理

3.1 检查并处理缺失值

3.2 删除无用的列

3.3 检查类别变量

3.4 标签编码

3.5 得到自变量和因变量

3.6 拆分训练集和测试集

3.7 特征缩放

4. 用不同参数构建AdaBoost回归模型

4.1 模型一

4.1.1 构建模型

4.1.2 测试

4.1.3 评估模型性能

4.2 模型二

4.2.1 建立模型

4.2.2 预测

4.2.3 评估模型性能

5 实验结论

0. 知识准备

询问chatgpt相关背景知识：

ChatGPT 4o

介绍一下adaBoost模型



AdaBoost (Adaptive Boosting, 自适应增强) 是一种集成学习方法，通过结合多个弱分类器来提高分类性能。AdaBoost的基本思想是将多个弱分类器组合成一个强分类器。以下是AdaBoost模型的详细介绍：

核心思想

AdaBoost的核心思想是通过逐步训练一系列的弱分类器，每个弱分类器在前一个弱分类器的基础上进行改进，使整体分类器的性能不断提升。

算法步骤

1. 初始化权重：
对每个训练样本赋予相同的初始权重 $w_i = \frac{1}{N}$ ，其中 N 是训练样本的数量。
2. 迭代训练：
AdaBoost重复进行以下步骤 T 次，每次迭代都会生成一个新的弱分类器。
 - 训练弱分类器：
使用当前样本权重训练一个弱分类器 h_t 。
 - 计算分类误差：
计算弱分类器在训练集上的加权误差。

1.导入包集

```
data = pd.read_csv('/home/dataset/BlackFriday.csv')
data
```

显示数据集:

```
[2]: data = pd.read_csv('/home/dataset/BlackFriday.csv')
data
```

```
[2]:
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	1000001	P00069042	F	0-17	10	A	2	0.0
1	1000001	P00248942	F	0-17	10	A	2	0.0
2	1000001	P00087842	F	0-17	10	A	2	NaN
3	1000001	P00085442	F	0-17	10	A	2	0.0
4	1000002	P00285442	M	55+	16	C	4+	0.0
...
49995	1001649	P00102642	M	18-25	19	C	2	1.0
49996	1001649	P00035842	M	18-25	19	C	2	1.0
49997	1001649	P00052842	M	18-25	19	C	2	1.0
49998	1001649	P00183142	M	18-25	19	C	2	1.0
49999	1001650	P00155642	M	26-35	19	C	1	0.0

50000 rows × 12 columns

3. 数据预处理

3.1 检查并处理缺失值

```
blank = data.isnull().sum()
blank
```

```
[5]: User_ID      0
      Product_ID  0
      Gender      0
      Age         0
      Occupation  0
      City_Category 0
      Stay_In_Current_City_Years 0
      Marital_Status 3
      Product_Category_1 0
      Product_Category_2 15721
      Product_Category_3 34817
      Purchase      0
      dtype: int64
```

```
data = data.drop(['Product_Category_2', 'Product_Category_3'], axis = 1)
```

```
blank = data.isnull().sum()
blank
```

```
[7]: User_ID      0
      Product_ID  0
      Gender      0
      Age         0
      Occupation  0
      City_Category 0
      Stay_In_Current_City_Years 0
      Marital_Status 3
      Product_Category_1 0
      Purchase      0
      dtype: int64
```

3.2 删除无用的列

```
data = data.drop(['User_ID', 'Product_ID'], axis = 1)
```

3.3 检查类别变量

```
print(data.dtypes)
```

```
[9]: print(data.dtypes)
```

```
Gender                object
Age                  object
Occupation            int64
City_Category         object
Stay_In_Current_City_Years  object
Marital_Status        float64
Product_Category_1    int64
Purchase              int64
dtype: object
```

```
# 转换变量类型
```

```
data['Stay_In_Current_City_Years'] =
data['Stay_In_Current_City_Years'].astype('int64')
data['Product_Category_1'] = data['Product_Category_1'].astype('object')
data['Occupation'] = data['Occupation'].astype('object')
data['Marital_Status'] = data['Marital_Status'].astype('object')
```

```
# 替换4+
```

```
data['Stay_In_Current_City_Years'].replace('4+', 4, inplace = True)
```

```
print(data.dtypes)
```

```
Gender                object
Age                  object
Occupation            object
City_Category         object
Stay_In_Current_City_Years  int64
Marital_Status        object
Product_Category_1    object
Purchase              int64
dtype: object
```

3.4 标签编码

```
data = pd.get_dummies(data, drop_first = True)
data
```

[14]:

	Stay_In_Current_City_Years	Purchase	Gender_M	Age_18-25	Age_26-35	Age_36-45	Age_46-50	Age_51-55	Age_55+	Occupation_1	...	Product_Cat
0	2	8370	0	0	0	0	0	0	0	0	...	
1	2	15200	0	0	0	0	0	0	0	0	...	
2	2	1422	0	0	0	0	0	0	0	0	...	
3	2	1057	0	0	0	0	0	0	0	0	...	
4	4	7969	1	0	0	0	0	0	1	0	...	
...
49995	2	1374	1	1	0	0	0	0	0	0	...	
49996	2	5372	1	1	0	0	0	0	0	0	...	
49997	2	18879	1	1	0	0	0	0	0	0	...	
49998	2	17029	1	1	0	0	0	0	0	0	...	
49999	1	6093	1	0	1	0	0	0	0	0	...	

50000 rows × 49 columns

3.5 得到自变量和因变量

```
# 得到自变量和因变量
y = data['Purchase'].values
data = data.drop(['Purchase'], axis = 1)
x = data.values
```

3.6 拆分训练集和测试集

```
# 拆分训练集和测试集
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
random_state = 1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(35000, 48)
(15000, 48)
(35000,)
(15000,)
```

3.7 特征缩放

```
# 特征缩放
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)
sc_y = StandardScaler()
y_train = np.ravel(sc_y.fit_transform(y_train.reshape(-1, 1)))
```

4. 用不同参数构建AdaBoost回归模型

4.1 模型一

4.1.1 构建模型

```
# 使用不同的参数构建AdaBoost回归模型
# 模型1
from sklearn.ensemble import AdaBoostRegressor
regressor = AdaBoostRegressor(n_estimators=50, learning_rate=1, loss='linear',
random_state=0)
regressor.fit(x_train, y_train)
```

```
[19]: AdaBoostRegressor(learning_rate=1, random_state=0)
```

4.1.2 测试

```
# 在测试集做预测
y_pred = regressor.predict(x_test)
y_pred[:5]
```

```
[20]: array([ 0.65004612,  0.18805047,  0.55246384,  0.55246384, -0.01299079])
```

```
# y_pred变回特征缩放之前的
y_pred = sc_y.inverse_transform(y_pred)
y_pred[:5]
```

```
[17]: array([12501.03102111, 10206.94045722, 12016.4753753 , 12016.4753753 ,
          9208.64781524])
```

4.1.3 评估模型性能

```
# 评估模型性能
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print("R2 Score:", r2)
```

```
R2 Score: 0.3075923325444474
```

4.2 模型二

4.2.1 建立模型

```
# 模型2
from sklearn.tree import DecisionTreeRegressor
regressor = AdaBoostRegressor(base_estimator =
DecisionTreeRegressor(min_samples_split=100, max_depth=10, min_samples_leaf=10),
```

```
n_estimators=1000, learning_rate=0.2,  
loss='linear', random_state=0)  
regressor.fit(x_train, y_train)
```

```
[19]: AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=10,  
                                                             min_samples_leaf=10,  
                                                             min_samples_split=100),  
                        learning_rate=0.2, n_estimators=1000, random_state=0)
```

4.2.2 预测

```
# 在测试集做预测  
y_pred = regressor.predict(x_test)  
y_pred[:5]
```

```
[24]: array([ 0.93492107, -1.37455773,  0.10419903,  0.08179523, -0.39398585])
```

```
y_pred = sc_y.inverse_transform(y_pred) # y_pred变回特征缩放之前的  
y_pred[:5]
```

```
[25]: array([13915.60913326, 2447.63636364, 9790.56682028, 9679.31829574,  
            7316.77459874])
```

4.2.3 评估模型性能

```
# 评估模型性能  
r2 = r2_score(y_test, y_pred)  
print("R2 Score:", r2)
```

R2 Score: 0.6070474824774648

5 实验结论

1. 使用AdaBoost可以解决回归问题

在本实验中，采用AdaBoost算法进行了回归分析。实验结果表明，AdaBoost不仅适用于分类问题，在处理回归问题上同样表现出色。通过集成多个弱回归模型，AdaBoost能够有效地提高预测的准确性和稳健性。实验结果显示，AdaBoost在处理复杂的非线性关系时表现出较强的适应能力，能够提供较高的预测精度。因此，可以确认AdaBoost是一种有效的回归方法，能够应用于各种回归任务中。

2. 不同超参数对模型性能的影响不同

实验进一步探讨了不同超参数设置对AdaBoost模型性能的影响。通过调整学习率、弱学习器数量等关键超参数，观察到模型的预测性能有显著变化。例如，适当增加弱学习器的数量可以提高模型的预测能力，但过多的学习器可能导致过拟合。同样，学习率的选择对模型收敛速度和预测精度也有显著影响。较高的学习率可能导致模型过拟合，而较低的学习率可能导致模型欠拟合。实验结果表明，优化超参数是提升AdaBoost回归模型性能的关键步骤，不同数据集和问题类型需要针对性地调整超参数设置，以获得最佳的模型性能。

通过上述实验，我们不仅验证了AdaBoost在回归问题上的有效性，还强调了超参数调优在模型训练中的重要性，为后续的应用和优化提供了宝贵的参考。