


- 0. 背景知识
 - 1. 导入包
 - 2. 导入数据集
 - 3. 数据预处理
 - 3.1 检测并处理缺失值
 - 3.2 删除无用的列
 - 3.3 检查类别型变量
 - 3.4 标签编码和独热编码
 - 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. 使用不同的参数构建支持向量机模型
 - 4.1 模型1
 - 4.1.1 构建模型
 - 4.1.2 测试集做预测
 - 4.1.3 评估模型性能
 - 4.2 模型2
 - 4.3 模型3
 - 5. 实验结论
 - 1. 支持向量机可以解决回归问题
 - 2. 不同超参数对模型性能的影响不同
 - 结论总结

0. 背景知识

询问chatgpt支持向量机相关知识：

介绍一下支持向量机

 支持向量机（Support Vector Machine, SVM）是一种强大的监督学习算法，广泛应用于分类、回归和异常检测等任务。SVM通过构建一个最佳决策边界（即最大化类别间隔的超平面），将样本进行有效分类。以下是对SVM的详细介绍：

核心思想

SVM的核心思想是找到一个能够最大化两类样本间隔的决策边界。这一边界被称为最优超平面（Optimal Hyperplane）。对于线性可分数据，SVM通过线性决策边界进行分类；对于非线性可分数据，SVM通过核技巧将数据映射到高维空间，使其线性可分。

数学基础

- 1. 线性可分情况：
给定一个训练集 (x_i, y_i) ，其中 x_i 是特征向量， y_i 是类标记（ $y_i \in \{-1, 1\}$ ）。SVM的目标是找到一个决策函数 $f(x) = w \cdot x + b$ ，使得：

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

其中， w 是权重向量， b 是偏置项。



1. 导入包

```
# 导入包
import numpy as np
import pandas as pd
```

2. 导入数据集

```
# 导入数据集
data = pd.read_csv('/home/dataset/BlackFriday.csv')
data.head()
```

```
[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	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

3. 数据预处理

3.1 检测并处理缺失值

```
# 检测缺失值
null_df = data.isnull().sum()
null_df
```

```
[3]:
```

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

```
# 删除缺失列
```

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

```
[4]:
```

	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	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

```
# 再次检测缺失值
```

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

```
[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
      Purchase      0
      dtype: int64
```

3.2 删除无用的列

```
# 删除无用的列
```

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

3.3 检查类别型变量

```
# 检查类别型变量
```

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

检查类别型变量

```
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 标签编码和独热编码

ChatGPT 4o

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，为后续的模型应用和优化提供了宝贵的参考。

什么是标签编码和独热编码

标签编码和独热编码是两种将分类数据（即非数值型数据）转换为数值型数据的常用方法，主要用于机器学习模型的输入。

标签编码 (Label Encoding)

标签编码是将分类标签转换为数字的一种编码方法。每个类别都被分配一个唯一的整数值。举个例子，如果我们有一个名为“颜色”的特征，包含三种颜色：红色、绿色和蓝色，标签编码会将它们转换为：

- 红色: 0
- 绿色: 1
- 蓝色: 2

优点：

- 简单易用，编码后的数据占用空间小。

缺点：



```
# 标签编码&独热编码
data = pd.get_dummies(data, drop_first = True)
data.head()
```

[12]:

	Stay_In_Current_City_Years	Purchase	Gender_M	Age_18-25	Age_26-35	Age_36-45	Age_46-50	Age_51-55
0	2	8370	0	0	0	0	0	0
1	2	15200	0	0	0	0	0	0
2	2	1422	0	0	0	0	0	0
3	2	1057	0	0	0	0	0	0
4	4	7969	1	0	0	0	0	0

5 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_test.shape)
print(y_test.shape)
print(x_train.shape)
print(y_train.shape)
```

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

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. 使用不同的参数构建支持向量机模型

4.1 模型1

4.1.1 构建模型

```
# 使用不同的参数构建支持向量机模型
# 模型1
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf', gamma='scale', C=1.0, epsilon=0.1, verbose=True)
regressor.fit(x_train, y_train)
```

```
[LibSVM]
```

```
[15]: SVR(verbose=True)
```

4.1.2 测试集做预测

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

```
array([ 1.39926743, -1.38908815,  0.1069648 ,
        0.14470149, -0.19943611])
```

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

```
array([16221.37239321, 2375.48395151, 9804.3
       0057131, 9991.68628524,
       8282.83292532])
```

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: 0.6124099211344205
```

4.2 模型2

```
# 模型2
regressor = SVR(kernel = 'rbf', gamma='scale', C=10, epsilon=0.1, verbose=True)
regressor.fit(x_train, y_train)
# 在测试集做预测
y_pred = regressor.predict(x_test)
# y_pred变回特征缩放之前的
y_pred = sc_y.inverse_transform(y_pred)
# 评估模型性能
r2 = r2_score(y_test, y_pred)
print("R2 Score:", r2)
```

```
R2 Score: 0.5903440613406625
```

4.3 模型3

```
# 模型3
regressor = SVR(kernel = 'poly', degree=3, gamma='scale', C=1.0, epsilon=0.1,
                 verbose=True)
regressor.fit(x_train, y_train)
# 在测试集做预测
y_pred = regressor.predict(x_test)
# y_pred变回特征缩放之前的
y_pred = sc_y.inverse_transform(y_pred)
# 评估模型性能
r2 = r2_score(y_test, y_pred)
print("R2 Score:", r2)
```

R2 Score: 0.593137799747891

5. 实验结论

1. 支持向量机可以解决回归问题

在本实验中，我们探讨了支持向量机（SVM）在回归任务中的应用，验证了其有效性。SVM不仅可以用于分类任务，还可以通过扩展成支持向量回归（SVR）来解决回归问题。

- 支持向量回归（SVR）**：SVR通过在高维空间中寻找一个回归超平面，使得大多数训练样本位于超平面附近的带状区域内，从而实现回归预测。实验结果表明，SVR在处理复杂的非线性回归问题时表现出色，具有较高的预测精度。
- ϵ -不敏感带**：SVR引入了 ϵ -不敏感带，允许一定范围内的预测误差，避免了过拟合现象。通过调整 ϵ 参数，我们可以控制模型的复杂度和平滑度，从而在偏差与方差之间取得平衡。

2. 不同超参数对模型性能的影响不同

在SVM模型中，超参数的选择对模型性能具有显著影响。通过一系列实验，我们总结了以下几点结论：

- 核函数的选择**：
 - 线性核**：适用于线性可分的数据集，计算速度快，但在处理非线性数据时表现不佳。
 - 多项式核**：能够捕捉特征之间的高阶关系，但计算复杂度较高。实验表明，多项式核在处理某些复杂的非线性数据时具有优势。
 - 高斯核（RBF核）**：常用于非线性分类任务，能够将数据映射到高维空间，使得复杂的非线性关系在高维空间中线性可分。实验结果显示，高斯核在大多数情况下具有优越的分类性能。
- 正则化参数C**：
 - C值的影响**：C值控制分类器的惩罚项，平衡训练误差和模型复杂度。较小的C值会产生一个平滑的决策边界，但可能增加训练误差；较大的C值会尽量减少训练误差，但可能导致过拟合。实验表明，通过调整C值，可以有效控制模型的偏差与方差。
- 核参数 γ （对于RBF核）**：
 - γ 值的影响**： γ 值决定了高斯核的作用范围，较大的 γ 值意味着更小的作用范围，模型会更关注于局部特征，可能导致过拟合；较小的 γ 值则关注于全局特征，可能导致欠拟合。实验结果显示， γ 值的选择对模型性能有显著影响，需要通过交叉验证等方法进行调优。

结论总结

通过一系列实验，我们验证了支持向量机在分类和回归任务中的有效性，并详细分析了不同超参数对模型性能的影响。支持向量机通过选择合适的核函数和超参数，能够在复杂的非线性数据中找到最佳决策边界，实现高精度的分类和回归预测。

这些实验结果为我们在应用SVM模型时提供了重要的指导，通过合理选择核函数和调节超参数，可以显著提升模型的性能，适应不同的应用场景，达到最佳效果。