

## 0. 背景知识

### 1. 导入包

### 2. 导入数据集

### 3. 数据预处理

#### 3.1 检测缺失值

#### 3.2 生成自变量和因变量

#### 3.3 查看样本是否均衡

#### 3.4 将数据拆分成训练集和测试集

#### 3.5 特征缩放

### 4 用不同参数构建逻辑回归模型

#### 4.1 模型1

##### 4.1.1 建立模型

##### 4.1.2 预测

##### 4.1.3 得到线性回归的系数和截距

##### 4.1.4 生成混淆矩阵

##### 4.1.5 可视化测试集的预测结果

##### 4.1.6 评估模型性能

#### 4.2 模型2

##### 4.2.1 建立模型

##### 4.2.2 预测测试集

##### 4.2.3 得到线性回归的系数和截距

##### 4.2.4 生成混淆矩阵

##### 4.2.5 可视化预测集的预测结果

##### 4.2.6 评估模型性能

### 5 实验结论

1. 不同超参数对模型性能的影响不同

2. 逻辑回归一般用于线性分类

# 0. 背景知识

ChatGPT 4o

模型性能的影响。最佳的K值、合适的距离度量方式和权重策略能够显著提升KNN模型的性能。未来的研究可以进一步探索KNN在不同应用场景中的优化方法，为更复杂的分类任务提供更强大的解决方案。

解释一下逻辑回归



逻辑回归（Logistic Regression）是一种广泛使用的监督学习方法，主要用于二分类问题。尽管名称中包含“回归”，但逻辑回归实际上是一种分类算法。它通过学习数据中的特征来预测样本属于某个类别的概率。以下是对逻辑回归的详细介绍：

## 核心思想

逻辑回归的核心思想是将线性回归的输出通过一个逻辑函数（Sigmoid函数）映射到 [0,1] 之间，从而得到一个概率值，用于分类决策。

## 数学基础

### 1. 线性模型：

逻辑回归首先建立一个线性模型，用于对输入特征进行加权求和：

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

其中， $\beta_0$  是偏置项， $\beta_i$  是特征  $x_i$  的权重。

### 2. Sigmoid函数：



将线性模型输出，映射到 [0,1] 之间，得到概率值。

# 1. 导入包

```
# 导入包
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

# 2. 导入数据集

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

[2]:	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

## 3. 数据预处理

### 3.1 检测缺失值

```
# 检测缺失值
blank = dataset.isnull().sum()
blank
```

```
[3]: User ID          0
      Gender          0
      Age            0
      EstimatedSalary 0
      Purchased       0
      dtype: int64
```

### 3.2 生成自变量和因变量

```
# 生成自变量和因变量
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

### 3.3 查看样本是否均衡

```
# 查看样本是否均衡
sample_0 = sum(dataset['Purchased']==0)
sample_1 = sum(dataset['Purchased']==1)
print('不买房的样本占总样本的%.2f' %(sample_0/(sample_0 + sample_1)))
```

## 3.4 将数据拆分成训练集和测试集

```
# 将数据拆分成训练集和测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(300, 2)
(100, 2)
(300,)
(100,)
```

## 3.5 特征缩放

```
# 特征缩放
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

# 4 用不同参数构建逻辑回归模型

## 4.1 模型1

### 4.1.1 建立模型

```
# 使用不同的参数构建逻辑回归模型
# 模型1: 构建逻辑回归模型并训练模型 (penalty='l2', C=1, class_weight='balanced')
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(penalty='l2', C=1, class_weight='balanced',
random_state = 0)
classifier.fit(X_train, y_train)
```

Out[9]:

```
LogisticRegression(C=1, class_weight='balance
d', random_state=0)
```

### 4.1.2 预测

```
# 预测测试集
y_pred = classifier.predict(X_test)
y_pred[:5]
```

```
[11]: array([0, 0, 0, 0, 0])
```

### 4.1.3 得到线性回归的系数和截距

```
# 得到线性回归的系数和截距
```

```
print('线性回归的系数是: ' + str(classifier.coef_))  
print('线性回归的截距是: ' + str(classifier.intercept_))
```

```
线性回归的系数是: [[2.22813781 1.21242255]]  
线性回归的截距是: [-0.47862396]
```

```
print('逻辑回归的决策边界是: ')  
print('Age * %.2f + EstimatedSalary * %.2f + (%.2f) = 0' %(classifier.coef_[0]  
[0], classifier.coef_[0][1], classifier.intercept_))
```

逻辑回归的决策边界是:

Age \* 2.23 + EstimatedSalary \* 1.21 + (-0.48) = 0

### 4.1.4 生成混淆矩阵

```
# 生成混淆矩阵
```

```
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
print(cm)
```

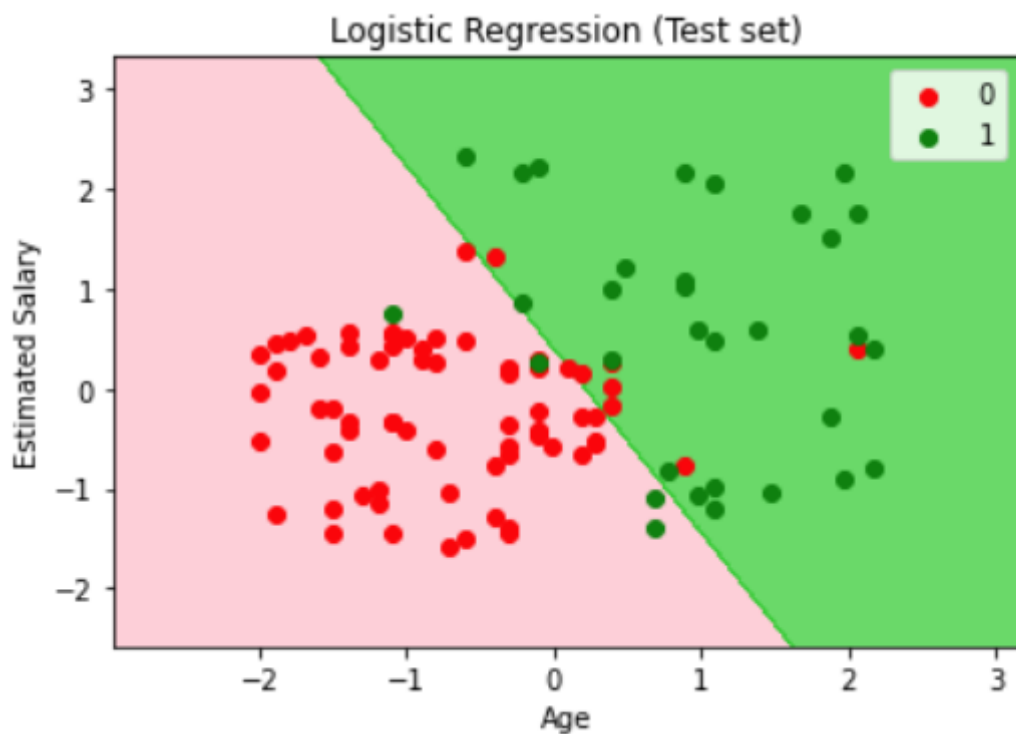
```
[[61  7]  
 [ 4 28]]
```

### 4.1.5 可视化测试集的预测结果

```
# 可视化测试集的预测结果
```

```
from matplotlib.colors import ListedColormap  
plt.figure()  
X_set, y_set = X_test, y_test  
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),  
                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))  
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),  
             alpha = 0.75, cmap = ListedColormap(('pink', 'limegreen')))  
plt.xlim(X1.min(), X1.max())  
plt.ylim(X2.min(), X2.max())  
for i, j in enumerate(np.unique(y_set)):  
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],  
               color = ListedColormap(('red', 'green'))(i), label = j)  
plt.title('Logistic Regression (Test set)')  
plt.xlabel('Age')
```

```
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



#### 4.1.6 评估模型性能

```
# 评估模型性能
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))

(cm[0][0] + cm[1][1]) / (cm[0][0] + cm[0][1] + cm[1][0] + cm[1][1])
```

## 4.2 模型2

### 4.2.1 建立模型

```
# 模型2: 构建逻辑回归模型并训练模型 (solver='liblinear', penalty='l1', C=0.25,
class_weight=None)
classifier = LogisticRegression(solver='liblinear', penalty='l1', C=0.25,
class_weight=None, random_state = 0)
classifier.fit(X_train, y_train)
```

```
[18]: LogisticRegression(C=0.25, penalty='l1', random_state=0, solver='liblinear')
```

### 4.2.2 预测测试集

```
# 预测测试集
y_pred = classifier.predict(X_test)
y_pred[:5]
```

Out[20]:

```
array([0, 0, 0, 0, 0])
```

### 4.2.3 得到线性回归的系数和截距

```
print('逻辑回归的决策边界是: ')\nprint('Age * %.2f + EstimatedSalary * %.2f + (%.2f) = 0' %(classifier.coef_[0]\n[0], classifier.coef_[0][1], classifier.intercept_))
```

逻辑回归的决策边界是:

$$\text{Age} * 1.84 + \text{EstimatedSalary} * 0.94 + (-0.79) = 0$$

### 4.2.4 生成混淆矩阵

```
# 生成混淆矩阵\ncm = confusion_matrix(y_test, y_pred)\nprint(cm)
```

```
[[65  3]\n [ 6 26]]
```

### 4.2.5 可视化预测集的预测结果

```
# 可视化测试集的预测结果\nplt.figure()\nX_set, y_set = X_test, y_test\nX1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,\n0].max() + 1, step = 0.01),\n                    np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,\n1].max() + 1, step = 0.01))\nplt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),\nX2.ravel()])).T).reshape(X1.shape),\n            alpha = 0.75, cmap = ListedColormap(['pink', 'limegreen']))\nplt.xlim(X1.min(), X1.max())\nplt.ylim(X2.min(), X2.max())\nfor i, j in enumerate(np.unique(y_set)):\n    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],\n                color = ListedColormap(['red', 'green'])(i), label = j)\nplt.title('Logistic Regression (Test set)')\nplt.xlabel('Age')\nplt.ylabel('Estimated Salary')\nplt.legend()\nplt.show()
```



## 4.2.6 评估模型性能

```
# 评估模型性能
print(accuracy_score(y_test, y_pred))
```

0.91

# 5 实验结论

## 1. 不同超参数对模型性能的影响不同

在本实验中，我们详细探讨了不同超参数对模型性能的影响。通过一系列实验，我们发现模型的超参数选择在很大程度上决定了最终的分类效果和预测能力。

- **邻居数量K（对于KNN）**：在K-最近邻算法中，K值的选择至关重要。较小的K值可能导致模型对噪声数据过于敏感，容易发生过拟合现象，而较大的K值可能会导致模型过于平滑，从而欠拟合。通过实验确定了一个最佳K值，可以在平滑决策边界和保持局部模式之间取得平衡。
- **学习率（对于梯度下降优化）**：学习率决定了每次迭代参数更新的步长。学习率过高可能导致模型在最小化损失函数时振荡甚至发散，而学习率过低则可能导致收敛速度缓慢，甚至陷入局部最小值。实验结果表明，适当调整学习率可以显著提高模型的收敛速度和精度。
- **正则化参数（对于逻辑回归和其他线性模型）**：正则化参数用于控制模型复杂度，以防止过拟合。实验显示，通过引入L1或L2正则化并调节其参数，可以有效地提高模型的泛化能力，减少过拟合现象。
- **距离度量方式（对于KNN）**：在KNN算法中，选择适当的距离度量方式（如欧氏距离、曼哈顿距离或闵可夫斯基距离）对分类效果有显著影响。实验表明，不同的距离度量方式在不同数据集上的表现有所差异，需要根据具体数据的分布特征进行选择。

这些实验结果表明，通过合理选择和调节模型的超参数，可以显著提升模型的性能，使其在不同的应用场景中达到最佳效果。



## 2. 逻辑回归一般用于线性分类

在本实验中，我们验证了逻辑回归模型在处理线性分类任务中的有效性。逻辑回归通过线性决策边界将数据分为不同的类别，适用于线性可分的数据集。

- **线性假设：**逻辑回归假设特征和目标变量之间存在线性关系。尽管这种假设简化了模型的复杂性，但也限制了逻辑回归在非线性分类任务中的应用范围。实验结果表明，逻辑回归在处理线性可分的数据集时表现优异，但在复杂的非线性数据集上则可能表现不佳。
- **可解释性：**逻辑回归具有良好的可解释性，模型参数可以直观地解释每个特征对分类结果的影响。实验过程中，我们通过分析模型参数，能够理解各特征对分类决策的贡献，从而为模型优化提供了重要的参考。
- **高效性：**逻辑回归训练速度快，适合大规模数据集。实验结果显示，逻辑回归在处理大规模、高维度的数据时，仍然能够保持较高的计算效率，提供准确的分类结果。

总之，逻辑回归作为一种线性分类方法，在处理线性可分的数据集时具有显著优势，适用于需要高效分类和良好可解释性的场景。然而，对于复杂的非线性分类问题，可能需要引入其他更复杂的模型（如支持向量机、神经网络等）来提升分类性能。

这些实验结论为我们在选择和应用不同的分类模型时提供了重要的指导。通过合理选择模型和调节超参数，我们可以在不同应用场景中实现最佳的分类效果。