**基于逻辑回归预测客户是否购买汽车新车型**

**1. 实验目的**

* 熟练掌握使用逻辑回归构建分类模型
* 熟练掌握逻辑回归模型的调参过程

**2. 实验内容**

国外换车比较普遍。每当该款汽车有新车型发布时，人们往往愿意购买。

某汽车公司有客户历史购车数据。现在该公司又推出一款新车型，请问销售人员如何从老客户中找到欲购买新车型的客户呢？

本实验通过逻辑回归模型预测顾客是否购买新车型。

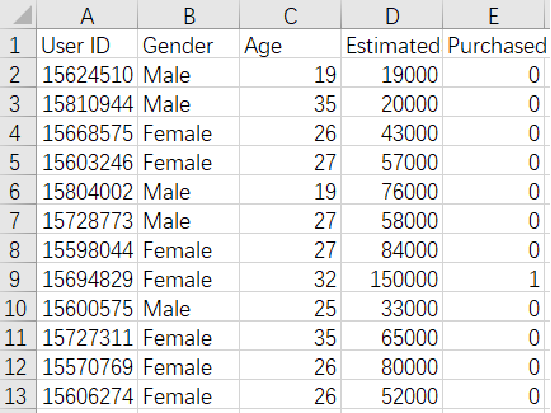
**3. 实验数据**

**数据集位置**

数据集存放在/home/dataset目录下。数据集名字是Social\_Network\_Ads.csv

**数据集格式**

某汽车公司有400条客户数据，包含客户ID、性别、年龄、工资、新车型发布时是否购买这5个字段。



**数据字段解释**

* User ID：字符串类型；用户ID，唯一标识用户
* Gender：字符串类型；性别；有2个值，分别是Female（女性）、Male（男性）
* Age：数值型；用户年龄
* Estimated：数值型；用户工资
* Purchased：字符串类型；新车型发布时是否购买；有2个值，分别是1（购买）、0（不购买）

**4. 实验知识点**

* 逻辑回归模型

**5. 实验时长**

1学时

**6. 实验环境**

* Linux Ubuntu 操作系统
* Jupyter 代码编辑器
* Python 3.6.9
* numpy 1.18.5
* pandas 1.1.5
* matplotlib 3.3.3
* scikit-learn 0.24.2

**7. 实验分析**

1. 导入包、导入数据、做数据预处理
2. 构建逻辑回归模型
3. 使用不同的参数，比较逻辑回归模型性能

**8. 实验过程**

In [1]:

*'''*

*基于逻辑回归预测客户是否购买汽车新车型*

*'''*

Out[1]:

'\n基于逻辑回归预测客户是否购买汽车新车型\n'

**8.1 导入包**

In [2]:

*# 导入包*

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**8.2 导入数据集**

In [3]:

*# 导入数据集*

dataset = pd.read\_csv('/home/dataset/Social\_Network\_Ads.csv')

dataset

Out[3]:

|  | **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

**8.3 数据预处理**

**8.3.1 检测缺失值**

In [4]:

*# 检测缺失值*

null\_df = dataset.isnull().sum()

null\_df

Out[4]:

User ID 0

Gender 0

Age 0

EstimatedSalary 0

Purchased 0

dtype: int64

**8.3.2 生成自变量和因变量**

为了可视化分类效果，仅选取 Age 和 EstimatedSalary 这2个字段作为自变量

In [5]:

*# 生成自变量和因变量*

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

**8.3.3 查看样本是否均衡**

如果样本不均衡，需要在调用模型类库时指明。否则模型性能指标不够客观。

In [6]:

*# 查看样本是否均衡*

sample\_0 = sum(dataset['Purchased']==0)

sample\_1 = sum(dataset['Purchased']==1)

print('不买车的样本占总样本的**%.2f**' %(sample\_0/(sample\_0 + sample\_1)))

不买车的样本占总样本的0.64

不买车的样本占总样本的0.64，表明样本基本是均衡的。

**8.3.4 将数据拆分成训练集和测试集**

In [7]:

*# 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,)

**8.3.5 特征缩放**

In [8]:

*# 特征缩放*

**from** **sklearn.preprocessing** **import** StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

**8.4 使用不同的参数构建逻辑回归模型**

**8.4.1 模型1**

参数 penalty='l2', C=1, class\_weight='balanced'

**8.4.1.1 构建逻辑回归模型并训练**

In [9]:

*# 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='balanced', random\_state=0)

**8.4.1.2 预测测试集**

In [10]:

*# 预测测试集*

y\_pred = classifier.predict(X\_test)

In [11]:

y\_pred[:5]

Out[11]:

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

**8.4.1.3 得到线性回归的系数和截距**

In [12]:

*# 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]

In [13]:

print('逻辑回归的决策边界是：')

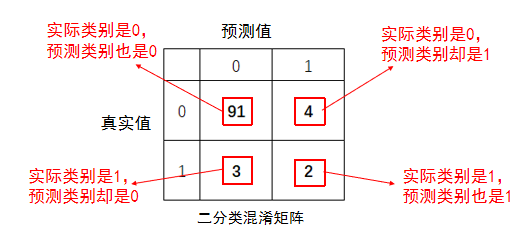
print('Age \* **%.2f** + EstimatedSalary \* **%.2f** + (**%.2f**) = 0' %(classifier.coef\_[0][0], classifier.coef\_[0][1], classifier.intercept\_) )

逻辑回归的决策边界是：

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

**8.4.1.4 生成混淆矩阵**

混淆矩阵的图解如下所示。



In [14]:

*# 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]]

由混淆矩阵可知，真实值是0（实际没买车），预测值也是0（预测没买车），这样的样本有61条；真实是0，预测是1，这样的样本有7条；真实值是1，预测是0，这样的样本有4条；真实值是1，预测值也是1，这样的样本有28条。

**8.4.1.5 可视化测试集的预测结果**

In [15]:

*# 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)

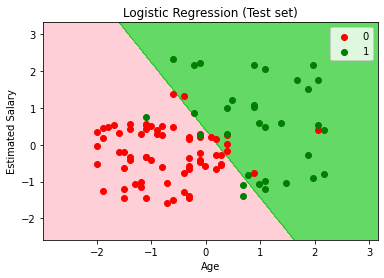
plt.title('Logistic Regression (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()



1. 红色点表示没有买车的样本点；
2. 绿色点表示买车的样本点；
3. 粉红色区域表示预测的不买车；
4. 草绿色区域表示预测的买车；
5. 大部分红色点都落在粉红色区域内，大部分绿色点都落在草绿色区域内，说明模型预测地较准确；
6. 逻辑回归做的是线性分类；

**8.4.1.6 评估模型性能**

可以调用函数库求出模型准确率

In [16]:

*# 评估模型性能*

**from** **sklearn.metrics** **import** accuracy\_score

print(accuracy\_score(y\_test, y\_pred))

0.89

也可以通过混淆矩阵求出模型准确率

In [17]:

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

Out[17]:

0.89

模型准确率不足0.9，待优化。

**8.4.2 模型2**

参数 solver='liblinear', penalty='l1', C=0.25, class\_weight=None

In [18]:

*# 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)

Out[18]:

LogisticRegression(C=0.25, penalty='l1', random\_state=0, solver='liblinear')

In [19]:

*# 预测测试集*

y\_pred = classifier.predict(X\_test)

In [20]:

y\_pred[:5]

Out[20]:

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

In [21]:

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 \* 1.84 + EstimatedSalary \* 0.94 + (-0.79) = 0

In [22]:

*# 生成混淆矩阵*

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

[[65 3]

[ 6 26]]

In [23]:

*# 可视化测试集的预测结果*

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)

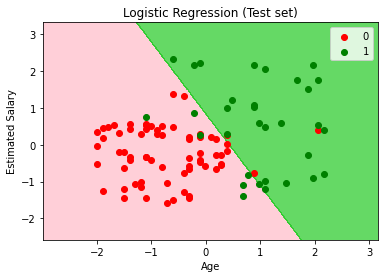
plt.title('Logistic Regression (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()



In [24]:

*# 评估模型性能*

print(accuracy\_score(y\_test, y\_pred))

0.91

现在模型准确率达到90%+了。模型2的性能比模型1要好。

**9. 实验结果（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；