第5天：逻辑回归

第1步：数据预处理 ¶

导入库

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

导入数据

In [2]:

```
dataset = pd.read_csv('../data/Social_Network_Ads.csv')
X = dataset.iloc[ : , [2,3]].values
Y = dataset.iloc[ : ,4].values
```

将数据集分成训练集和测试集

In [3]:

```
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)
```

特征缩放

In [4]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

第2步：逻辑回归模型

将逻辑回归应用于训练集

In [5]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, Y_train)
```

Out[5]:

```
LogisticRegression()
```

第3步：预测结果

In [6]:

```
Y_pred = classifier.predict(X_test)
Y_pred
```

Out[6]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

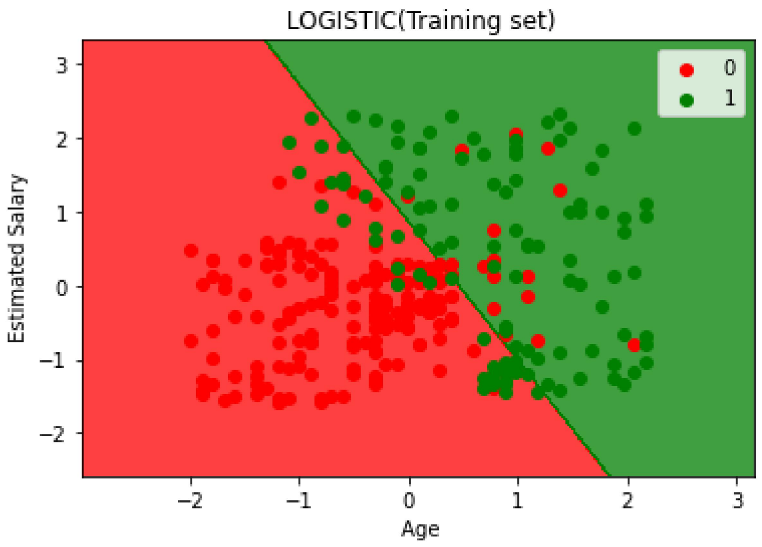
第4步：评估预测结果

可视化

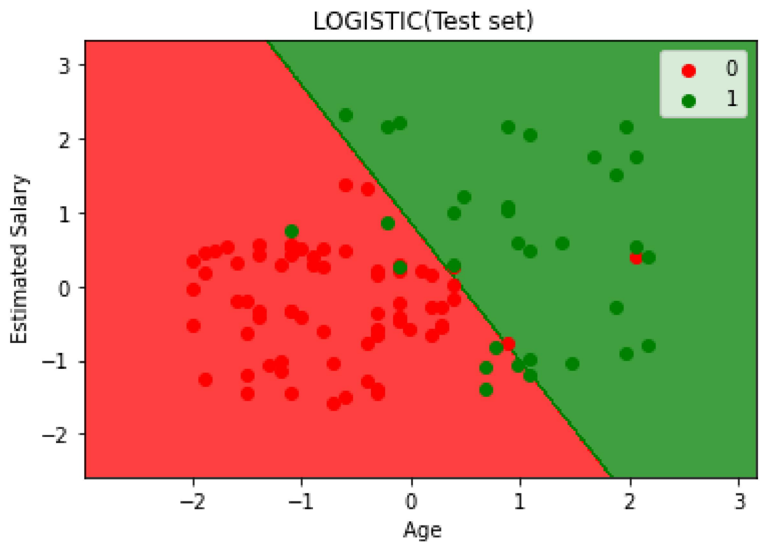
In [7]:

```
from matplotlib.colors import ListedColormap
X_set,y_set=X_train,Y_train
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(('red', 'green')))
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],
                c = ListedColormap(('red', 'green'))(i), label=j)

plt. title(' LOGISTIC(Training set)')
plt. xlabel(' Age')
plt. ylabel(' Estimated Salary')
plt. legend()
plt. show()

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(('red', 'green')))
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],
                c = ListedColormap(('red', 'green'))(i), label=j)

plt. title(' LOGISTIC(Test set)')
plt. xlabel(' Age')
plt. ylabel(' Estimated Salary')
plt. legend()
plt. show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
ided as value-mapping will have precedence in case its length matches with *x* & *
y*.  Please use the *color* keyword-argument or provide a 2-D array with a single
row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
ided as value-mapping will have precedence in case its length matches with *x* & *
y*.  Please use the *color* keyword-argument or provide a 2-D array with a single
row if you intend to specify the same RGB or RGBA value for all points.
```



```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
ided as value-mapping will have precedence in case its length matches with *x* & *
y*.  Please use the *color* keyword-argument or provide a 2-D array with a single
row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
ided as value-mapping will have precedence in case its length matches with *x* & *
y*.  Please use the *color* keyword-argument or provide a 2-D array with a single
row if you intend to specify the same RGB or RGBA value for all points.
```



```
In [ ]:
```