```
Logistic_Regression
```

```
Import Libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
```

Importing dataset and splitting dataset into training and test sets

```
pd = pd.read_csv('Social_Network_Ads.csv')
X = pd.iloc[:, :-1].values
y = pd.iloc[:, -1].values
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Feature Scaling

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Training the Logistic Regression model on the training set

Predicting a new result

```
print(classifier.predict(sc.transform([[30,87000]])))
```

→ [0]

Predicting the test set result

y_predict = classifier.predict(x_test)

```
 \begin{split} \text{display(y\_predict)} \\ & \xrightarrow{} \text{array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1]) \\ \end{aligned}
```

print(np.concatenate((y_predict.reshape(len(y_predict),1), y_test.reshape(len(y_test),1)),1))

```
[[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]
[1 0]
[0 0]
```

[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [1 1] [0 0] [0 0] [1 1] [0 0] [1 1] [0 0] [1 1] [0 0] [0 0] [0 0] [0 0] [0 1] [1 1] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [1 1] [0 0] [0 0] [0 0] [0 0] [1 1] [0 0] [0 0] [1 1] [0 0] [1 1] [1 1] [0 0] [0 0]

Making the confusion matrix

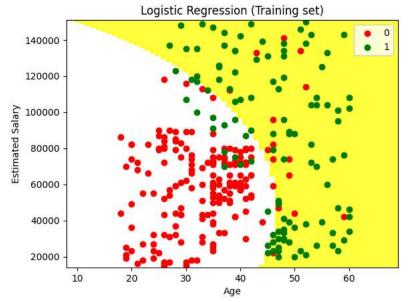
[1 0] [1 1] [0 1] [0 0] [0 0]

```
confusion_matrix(y_test, y_predict)
```

accuracy_score(y_test, y_predict)

→ 0.9

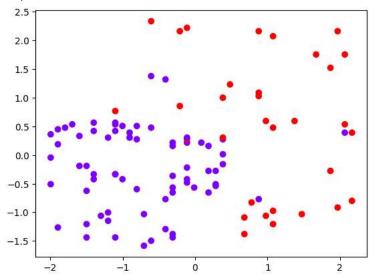
<ipython-input-14-ac20f31602f7>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided
plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)



Visualising the Test set results

plt.scatter(x_test[:, 0], x_test[:, 1], c = y_test, cmap = 'rainbow')





<ipython-input-16-edea852c2b2b>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided
plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)

