```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# importing dataset
dataset = pd.read_csv("Social_Network_Ads.csv")
X = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values



#3   (splitting dataset into train_test_split)
#   Splitting the dataset into the Training set and Test set
#  Totally 400 rows of data and 300 for training set and 100 for test_set

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)



# feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X =  StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test  = sc_X.transform(X_test)


# fitting Logestic regression to the training dataset
from sklearn.linear_model import LogisticRegression
#    There are many optional parameters. Lets only use random_state=0
#    We create a classifier object of LR class

classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train,y_train)

# predicting the test set result
y_pred = classifier.predict(X_test)



# (evaluting the model performance)
#   Making the Confusion Matrix. It contains the correct and #incorrect predictions of our model
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred) ## confusion matrix [tP,FP
                                     #                   FP,TN])

# visualizing the trannig set:
#    ListedColormap class help us to colorize the data points.
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 =
                    np.arange(start = X_set[:,1].min() - 1, stop = X_set[:,1].max() + 1, step  =


# ( X_set[:,0].min() - 1 {select min.value from col 0, then  adding -1}
#   X_set[:,0].max()+1)  {select max().value from col 0 then adding +1}
#   X_Set use with minus 1 and plus 1 to prevent ponits to be squeezed #on the axes.
```

```
#   Create the grid. step=0.01 means all the pixels were actually with  0.01 resolution. min and
#   X_Set use with minus ana plus one to prevent ponits to be squeezed on the axes.

#############################################################################################
#  This is the line applying the classifier on all the pixel #observation points. It colors all
#  points and the blue pixel points. contour function make the contour #between red and blue reg
plt.contourf(X1,X2,classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red','green')))


#plot the limits of the age and the estimated salary lines.
plt.xlim(X1.min() , X1.max())
plt.ylim(X2.min() , X2.max())

#This loop here plots all the data points that are the real values.

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) ,la
     # show scatter plot

#Add the name of the plot and the labels.
plt.title('Logistic Regression(Training Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

#############################################################################
# listedColormap to provide red and green color
#
```