In [1]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import math
from sklearn.model_selection import train_test_split
```

In [28]:
```
def sigmoid_function(X):
  return 1/(1+math.e**(-X))

def log_regression4(X, y, alpha, epochs):
  y_ = np.reshape(y, (len(y), 1)) # shape (150,1)
  N = len(X)
  theta = np.random.randn(len(X[0]) + 1, 1) #* initialize theta
  X_vect = np.c_[np.ones((len(X), 1)), X] #* Add x0 (column of 1s)
  avg_loss_list = []
  loss_last_epoch = 9999999
  for epoch in range(epochs):
    sigmoid_x_theta = sigmoid_function(X_vect.dot(theta)) # shape: (150,5).(
    grad = (1/N) * X_vect.T.dot(sigmoid_x_theta - y_) # shapes: (5,150).(150
    best_params = theta
    theta = theta - (alpha * grad)
    hyp = sigmoid_function(X_vect.dot(theta)) # shape (150,5).(5,1) = (150,1
    avg_loss = -np.sum(np.dot(y_.T, np.log(hyp) + np.dot((1-y_).T, np.log(1-
    # if epoch % 50 == 0:
    #   print('epoch: {} | avg_loss: {}'.format(epoch, avg_loss))
    #   print('')
    avg_loss_list.append(avg_loss)
    loss_step = abs(loss_last_epoch - avg_loss) #*
    loss_last_epoch = avg_loss #*
    # if loss_step < 0.001: #*
    #   # print('\nStopping training on epoch {}/{}, as (last epoch loss - c
    #   break #*
  # plt.plot(np.arange(1, epoch+1), avg_loss_list[1:], color='red')
  # plt.title('Cost function')
  # plt.xlabel('Epochs')
  # plt.ylabel('Cost')
  # plt.show()
  return best_params
```

In [29]:
```
best_params
```

Out[29]:
```
array([[-0.79195493],
       [ 0.31465538],
       [ 1.07877386],
       [-0.26848734],
       [ 0.07104752],
       [-0.16642498],
       [ 0.69551999],
       [ 0.29832286],
       [ 0.23943724]])
```

In [31]:
```
# Load Dataset
#iris = datasets.load_iris()
# Define X features
#X = iris["data"]
# Define binary target 'y' based on iris plant type
#y_seto = (iris["target"] == 0).astype(int) # return 1 if Iris Setosa (0 = s
#y_vers = (iris["target"] == 1).astype(int)
#y_virg = (iris["target"] == 2).astype(int)
# List of ys
#y_iris_types = [y_seto, y_vers, y_virg]
#y_iris_types = {'Setosa':y_seto,
#                'Versicolor':y_vers,
#                'Virginica':y_virg}
#predicted_probs = {'Setosa':0.0,
#                   'Versicolor':0.0,
#                   'Virginica':0.0}
#actual_y = {'Setosa':0,
#             'Versicolor':0,
#            'Virginica':0}
import numpy as np
import pandas as pd
df = pd.read_csv('diabetes.csv')

X = df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','E
y = df[['Outcome']]
X = df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','E

y_no = (df["Outcome"] == 0).astype(int)
y_yes = (df["Outcome"] == 1).astype(int)

y_diabetes_types = [y_no, y_yes]
y_diabetes_types = {'No':y_no,
                'Yes':y_yes}
predicted_probs = {'No':0.0,
                   'Yes':0.0}
actual_y = {'No':0,
            'Yes':0}

for key, y_diabetes_type in y_diabetes_types.items():
  # Split dataset (training and testing sets)
  X_train, X_test, y_train, y_test = train_test_split(X, y_diabetes_type, te
  # Scale X
  sc = StandardScaler()
  X_train = sc.fit_transform(X_train)
  X_test = sc.transform(X_test)
  # Train model
  epochs = 1000
  alpha = 0.01
  best_params = log_regression4(X_train, y_train, alpha, epochs)
  # Make predictions on test set
  index_ = 10
  X_to_predict = [list(X_test[index_])]
  #print(X_to_predict)
  X_to_predict = np.c_[np.ones((len(X_to_predict), 1)), X_to_predict] # add
  #print(X_to_predict)
```

```
    pred_probability = sigmoid_function(X_to_predict.dot(best_params))
    #print(pred_probability)
    predicted_probs[key] == pred_probability[0][0]
    print('Our model calculated probability of sample being {}, is: {}%'.forma
    actual_y[key] = y_test[index_]

max_key = max(predicted_probs, key=predicted_probs.get)
print('\n', predicted_probs)
print('\nModel Prediction: {}'.format(max_key))
max_actual_y = max(actual_y, key=actual_y.get)
print('Real value is: {}'.format(max_actual_y))
```

```
Our model calculated probability of sample being No, is: 40.65%
Our model calculated probability of sample being Yes, is: 83.27%

 {'No': 0.0, 'Yes': 0.0}

Model Prediction: No
Real value is: No
```

In [ ]:

In [ ]:

Resumen

Se utilizó la librería sci kit learn para hacer el split para las nuevos data frames de test y train, también se utilizó para realizar normalizar y ajustar los datos, sin embargo no se utilizó para cuestiones de predicción. Lo que se hizo en este programa, primero se definió la función sigmoide que es útil para clasificar a nuestros datos, después se implementó la función logregresion4 para obtener los mejores parámetros que van a afectar a nuestro data Frame de Test. Posterior a esto se realizó la lectura de los bases de datos y dividir las features y el label, esto para poder introducir los datos correctamente a el ciclo for que prueba los valores, best_params que fueron entregados por logregresion4, y que contiene los parámetros que minimizan el avg_loss. Se probó con diferentes valores de epoch y alpha, los mejores valores fueron 1000 y 0.01 respectivamente.