Question 1

Logistic Regression for multiclass classification

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [2]:
```
dataset = pd.read_csv('Iris.csv', header=0)
dataset.sample(n=5)
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
59	60	5.2	2.7	3.9	1.4	Iris-versicolor
4	5	5.0	3.6	1.4	0.2	Iris-setosa
84	85	5.4	3.0	4.5	1.5	Iris-versicolor
53	54	5.5	2.3	4.0	1.3	Iris-versicolor
45	46	4.8	3.0	1.4	0.3	Iris-setosa

In [3]:
```
x = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
x = np.hstack((np.ones((x.shape[0], 1)), x)) #[1 x]
print(x.shape)
```

```
(150, 6)
```

In [4]:
```
y_class = y.unique()
y_class
```

Out[4]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [5]:
```
#in the given dataset, replace the Species column with the numerical value that refe
Y = np.zeros((y.shape[0], len(y_class)))
print(Y.shape)
for i in range(len(Y)):
    for j in range(len(y_class)):
        if y_class[j] == y[i]:
            Y[i][j] = 1
```

```
(150, 3)
```

In [6]:
```
train_x, test_x, train_y, test_y = train_test_split(x, Y, train_size=0.8, shuffle=Tr
```

In [7]:
```
print(train_x.shape)
print(test_x.shape)
print(train_y.shape)
print(test_y.shape)
```

```
(120, 6)
(30, 6)
```

```
(120, 3)
(30, 3)
```

In [8]:
```
theta = np.zeros((x.shape[1], len(y_class)))
theta.shape
```

Out[8]:
```
(6, 3)
```

In [9]:
```
def sigmoid(x, theta):
    return 1/(1+np.exp(-np.dot(x, theta)))

def multiclass_logistic_regression(x, y, theta, alpha, iterations):
    m = x.shape[0]
    for _ in range(iterations):
        prec_y = sigmoid(x, theta)
        theta = theta - (alpha/m)*np.dot(x.T, prec_y - y)
    return theta
```

In [10]:
```
theta = multiclass_logistic_regression(train_x, train_y, theta, 0.0002, 70000)
```

In [11]:
```
prediction = sigmoid(test_x, theta)
for i in prediction:
    ind = np.where(i == np.amax(i))
    for j in range(len(i)):
        i[j] = 1 if ind[0][0] == j else 0
prediction
```

Out[11]:
```
array([[0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.]])
```

In [12]:
```
error = (prediction == test_y)
```

```
error
```

Out[12]:
```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True, False, False],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True, False, False],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True],
       [ True, False, False],
       [ True,  True,  True]])
```

In [13]:
```
s = len(y_class)
error_prec = 0
for i in error:
    if np.sum(i) != s:
        error_prec += 1
print(error_prec)
```

```
3
```

In [14]:
```
percentage_error = (error_prec/len(prediction))*100
percentage_error
```

Out[14]:
```
10.0
```

In [15]:
```
print(f"Accuracy: {100-percentage_error}")
```

```
Accuracy: 90.0
```

In [ ]: