DS Lab Assignment 6

Kadali Sai Vivek

197139

1) Logistic Regression for multiclass classification from scratch

In [66]:

```
#Importing Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

data = pd.read_csv('Iris.csv')
data
```

Out[66]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [67]:

```
#Splitting into Training and Testing Data
X, y = data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']], data['

Y=[];
for i in y:
    if i == 'Iris-setosa':
        Y.append(0)
    if i ==  'Iris-versicolor':
        Y.append(1)
    if i == 'Iris-virginica':
        Y.append(2)

Y=np.array(Y)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

In [68]:

```
# Defining Logistic Regression
class LogisticRegression:

    def __init__(self, Learning_rate=0.1, Num_of_iterations=10000):
        self.Learning_rate = Learning_rate
        self.Num_of_iterations = Num_of_iterations
        self.weights = None

    def fit(self,X,y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)

        for _ in range(self.Num_of_iterations):
            linear_model = X @ self.weights
            hx = self._sigmoid(linear_model)

            dw = (X.T * (hx - y)).T.mean(axis=0)
            db = (hx - y).mean(axis=0)

            self.weights -= self.Learning_rate * dw

    def predict(self,X):
        linear_model = np.dot(X,self.weights)
        y_predicted = self._sigmoid(linear_model)
        return y_predicted

    def _sigmoid(self,x):
        return(1/(1+np.exp(-x)))
```

In [69]:

```
# Defining Multiclass Classification Using the Logistic Regression
class MulticlassClassification:

    def __init__(self):
        self.models = []

    def fit(self, X, y):
        for y_i in np.unique(y):

            x_true = X[y == y_i]
            x_false = X[y != y_i]
            x_true_false = np.vstack((x_true, x_false))

            y_true = np.ones(x_true.shape[0])
            y_false = np.zeros(x_false.shape[0])
            y_true_false = np.hstack((y_true, y_false))

            model = LogisticRegression()
            model.fit(x_true_false, y_true_false)
            self.models.append([y_i, model])
    def predict(self, X):
        y_pred = [[label, model.predict(X)] for label, model in self.models]
        output = []
        for i in range(X.shape[0]):
            max_label = None
            max_prob = -10000
            for j in range(len(y_pred)):
                prob = y_pred[j][1][i]
                if prob > max_prob:
                    max_label = y_pred[j][0]
                    max_prob = prob
            output.append(max_label)

        return output
```

In [70]:

```
#Using MulticlassClassification Class defines
model = MulticlassClassification()

model.fit(X_train, y_train)

y_pred=model.predict(X_test)
```

In [71]:

```
#Calculating Error
total=len(y_pred)
no_of_matched=0;
for i in range(len(y_pred)):
    if y_pred[i] == y_test[i]:
        no_of_matched+=1;

Accuracy=(no_of_matched)/(total)
print("Accuracy is :",Accuracy)
print("Error of the model is : ",1-Accuracy)
```

```
Accuracy is : 0.9666666666666667
Error of the model is :  0.033333333333333326
```