Machine Learning Assignment 2

Name: Khadija Sitabkhan
Student Id: 20236001
Course: 1CSD1 MSc Data Analytics

Assignment Details

Algorithm:

Logistic regression Algorithm.

Programming Language:

Python (3.8.6)

IDE:

PyCharm 2020.2.4 (Community Edition)

Description:

Logistic Regression is a supervised learning classification algorithm. We know that linear regression algorithm, outputs a continuous set of values. On the contrary, logistic regression uses the sigmoid function to return a probability which is then mapped to two or more discrete classes.

Types of Logistic regression algorithms:

I. **Binary logistic regression** is the statistical technique used to predict the relationship between the dependent variable (Y) and the independent variable (X), where the dependent variable is binary in nature. For example, the output can be Success/Failure, 0/1, True/False, or Yes/No

II. **Multinomial logistic regression** is used when you have one categorical dependent variable with two or more unordered levels (i.e two or more discrete outcomes). It is very similar to logistic regression except that here you can have more than two possible outcomes. For example, let's imagine that you want to predict what will be the most-used transportation type in the year 2030. The transport type will be the dependent variable, with possible outputs of train, bus, tram, and bike (for example).

III. **Ordinal logistic regression** is used when the dependent variable (Y) is ordered (i.e., ordinal). The dependent variable has a meaningful order and more than two categories or levels. Examples of such variables might be t-shirt size (XS/S/M/L/XL), answers on an opinion poll (Agree/Disagree/Neutral), or scores on a test (Poor/Average/Good).

1. Assumptions:

- Sample size must be large.
- The variables must be independent of each other. For example, in our data set we have variables like calorific value and colour. Hence these variables must be independent of each other.

2. Decisions:

- The value of learn rate (alpha) is 0.03
- Number of Iterations: 10000
- Solver for sklearn logistic regression: lbfgs

3. Sigmoid function:

In order to map predicted values to probabilities, we use the sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

Math:

$$S(z)=1/(1+e^{-z})$$

s(z) = output between 0 and 1 (probability estimate)

z = input to the function (your algorithm's prediction e.g. mx + b)

e = base of natural log

4. **Cost:**

Also known as the Cross-Entropy or log loss function. the cost function penalizes confident and wrong predictions more than it rewards confident and right predictions. It measures the performance of a classification model whose output is a probability value between 0 and 1

Math:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

m -> number of observations

y -> the predicted array

h -> sigmoid value of dot product of X and theta (the value of Z is between 0 and 1)

5. **Gradient Descent:**

Our goal is to minimize the loss function and the way we have to achieve it is by increasing/decreasing the weights, i.e. fitting them.

Math:

$$Gradiant\left(\delta \frac{(p(z))}{dz}\right) = \frac{1}{m} X\,(\,p(z) - y\,)$$

m -> length of oservations

X-> dot product of transpose of X

p(z) -> sigmoid value

6. **Preprocessing:**

We have implemented a function to normalise the data in our X variable. This uses the standard normalisation formula:

$$\mathbf{(X-X_{mean})/std}$$

X-> individual value

$X_{mean\ ->}$ mean of values in the column

Std -> standard deviation of the column

Algorithm:

Overview of the program

1. Load data into dataframe from beer.txt
2. Separate X and Y from loaded DF
3. Normalise the data in X .
4. Split the data into training and testing set using train_test_split function available in sklearn
5. Train the model using the training Data set to generate value of theta
6. Using theta predict the value of Y using the test data set(x_test) and store it in Y_predicted.
7. Compare the y_predicted and y_test to calculate accuracy of the model
8. Use the same X_test in step 6 generate Y_predicted using logistic regression algorithm from sklearn package
9. Compare y_predicted and y_test generated in step 8 to calculate accuracy.
10. Compare the accuracy of the implemented algorithm and the one present in the sklearn package.

11. Generate graphs of the cost function.

Logistic Regression Algorithm for multiclass

Explaining Step 5 from the overview above in detail.

1. Implement the Ove-vs-all classifier to generate discrete Y values based on the number of classes available.
2. Implement the binary classifier making pairs as below to generate the value of sigmoid , gradient_descent and cost
    a. Set1 = ale; set 2 : stout+ lager
    b. Set 1 = lager; set 2 : stout+ ale
    c. Set 1 = stout ; set 2 : ale + lager
3. Details of the binary classifier:
    a. Generate sigmoid(h) using formula described above from dot product of X & theta.
    b. Generate gradient_value value using formula :
        np.dot(X.T, (h - y_onevsall)) / length_pred
    c. Update the value of theta as the difference between theta and the product of learning_rate & gradient_value.
    d. Calculate the value of cost using the formula above.
4. Use the X_test set generated in step 4 in overview described above. The value of theta creates the decision boundaries depending on the values of Y predicted as to which class it needs to belong to.
5. We use the maximum probability to assign a value of Y.

Advantages of logistic regression:

- Logistic regression is much easier to implement than other methods, especially in the context of machine learning.
- Logistic regression works well for cases where the dataset is linearly separable.
- Logistic regression provides useful insights.

Disadvantages of logistic regression:

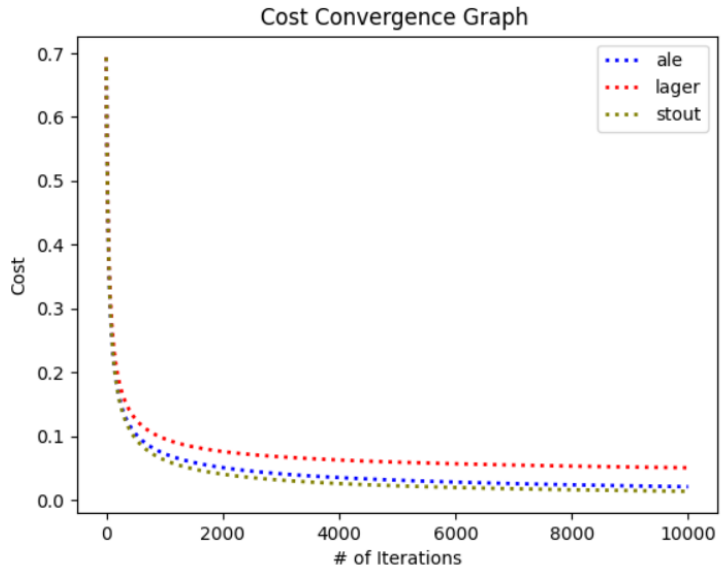
- Logistic regression fails to predict a continuous outcome.
- Logistic regression assumes linearity between the predicted (dependent) variable and the predictor (independent) variables.
- Logistic regression may not be accurate if the sample size is too small.

Conclusion:

Logistic Regression is the popular way to predict the values if the target is binary or ordinal. Only the requirement is that data must be clean and no missing values in it.

Plot of Cost:

The cost (or loss) function measures the difference, or error, between actual y and predicted y at its current position. This improves the machine learning model's efficacy by providing feedback to the model so that it can adjust the parameters to minimize the error and find the local or global minimum. It continuously iterates (according to the number of iterations we have provided), moving along the direction of steepest descent (or the negative gradient) until the cost function is close to or at zero. At this point, the model will stop learning..



Accuracy Comparison:

Generating report for accuracy of multiple iterations of Logistic algorithm with different samples.
----------Iteration # 1 ----------
The accuracy of the developed model is 94.12 %
Accuracy of Logistic Regression using sklearn is : 92.16 %
----------Iteration # 2 ----------
The accuracy of the developed model is 100.0 %
Accuracy of Logistic Regression using sklearn is : 100.0 %
----------Iteration # 3 ----------
The accuracy of the developed model is 98.04 %
Accuracy of Logistic Regression using sklearn is : 98.04 %
----------Iteration # 4 ----------
The accuracy of the developed model is 94.12 %
Accuracy of Logistic Regression using sklearn is : 94.12 %
----------Iteration # 5 ----------
The accuracy of the developed model is 96.08 %
Accuracy of Logistic Regression using sklearn is : 96.08 %
----------Iteration # 6 ----------
The accuracy of the developed model is 96.08 %
Accuracy of Logistic Regression using sklearn is : 96.08 %
----------Iteration # 7 ----------
The accuracy of the developed model is 100.0 %
Accuracy of Logistic Regression using sklearn is : 100.0 %
----------Iteration # 8 ----------
The accuracy of the developed model is 96.08 %
Accuracy of Logistic Regression using sklearn is : 96.08 %
----------Iteration # 9 ----------
The accuracy of the developed model is 96.08 %
Accuracy of Logistic Regression using sklearn is : 94.12 %
----------Iteration # 10 ----------
The accuracy of the developed model is 96.08 %
Accuracy of Logistic Regression using sklearn is : 96.08 %

Hence, we can conclude that the accuracy for the developed model is more or less similar to the sklearn model.

References:

1. https://machinelearningmastery.com/logistic-regression-tutorial-for-machine-learning/
2. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

3. https://www.ibm.com/cloud/learn/gradient-descent#:~:text=Gradient%20descent%20is%20an%20optimization%20algorithm%20which%20is,its%20accuracy%20with%20each%20iteration%20of%20parameter%20updates

Appendix:

```
'''
Machine Learning Assignment 2
Goal : select, implement and evaluate a machine learning algorithm.
Algorithm Used : Logistic Regression
Name: Khadija Sitabkhan
Course: 1CSD1 MSc Data Analytics
'''

#Importing libraries
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import sys

#class for all functions that belong to the algorithm developed.

class Logistic_Multilinear(object):
    #constructor function to initialise all variables local to the class object.
    #the value or alpha or learning_rate as 0.03 and number of iterations as 100.
    def __init__(class_obj, learn_rate=0.03, n_iteration=100):
        class_obj.theta = []
        class_obj.learn_rate = learn_rate
        class_obj.n_iter = n_iteration
        class_obj.cost = []

    def sigmoid(class_obj, x):
        value = 1 / (1 + np.exp(-x))
        return value

    # This function is used to train the data set and prepare the value of theta and cost
    # this value for theta and cost generated is used to predict the output of the test
    data set.

    def train_algo(class_obj, X, y):
        X = np.insert(X, 0, 1, axis=1)
        length_pred = len(y)

        # since we have 3 classes here the one vs all will iterate 3 times. for ale,
        lager and stout.

        for i in np.unique(y):
            cost1 = []
            y_onevsall = np.where(y == i, 1, 0)
            theta = np.zeros(X.shape[1])
            for val in range(class_obj.n_iter):
                z = X.dot(theta)
                h = class_obj.sigmoid(z)
                gradient_value = np.dot(X.T, (h - y_onevsall)) / length_pred
                theta -= class_obj.learn_rate * gradient_value

                r = len(y_onevsall)

                cost1.append((1 / r) * (np.sum(-y_onevsall.T.dot(np.log(h)) - (1 -
                y_onevsall).T.dot(np.log(1 - h)))))
            #the value and theta and cost is saved against each class.
            #3 here. (For ale, lager, stout)
            class_obj.theta.append((theta, i))
            class_obj.cost.append((cost1, i))
        return class_obj

    #This function calculates the y_predicted from X test set based on the value of
    theta from our training set
    def test_algo(class_obj, X):
        X = np.insert(X, 0, 1, axis=1)
```

```
64          Y_predicted = [max((class_obj.sigmoid(i.dot(theta)), c) for theta, c in
            class_obj.theta)[1] for i in X]
65          return Y_predicted
66
67      #this function calculates all the correctly predicted value over the total number
        of values.
68      #we are using this only for our algorithm.
69      def accuracy_functn(class_obj, Y_predicted, y):
70          score = sum(Y_predicted == y) / len(y)
71          return score
72
73      #function to plot the cost of all 3 classes
74      def plot_cost(class_obj, costh):
75          colour_plot = {'ale': "blue", 'lager': "red", 'stout': "olive"}
76
77          for cost, c in costh:
78              plt.plot(range(len(cost)), cost, 'r',
79                       color=colour_plot[c], label=c, linewidth=2, linestyle='dotted')
80          plt.title("Cost Convergence Graph ")
81          plt.xlabel("# of Iterations")
82          plt.ylabel("Cost")
83          plt.legend()
84          plt.show()
85
86  #scalar function is for preprocessing the data.
87  # we are normalising the data in X to return values between 0 & 1.  Using the formula:
88  # (X- Xbar)/Std where Xbar is the mean of the entire column and std is the standard
    deviation of the values in the column
89  def Scalar_function(X):
90      scale = X.copy()
91      for j in range(X.shape[1]):
92          mean = np.mean(X.iloc[:, j])
93          sd = np.std(X.iloc[:, j])
94          for i in range(X.shape[0]):
95              scale.iloc[i, j] = (X.iloc[i, j] - mean) / sd
96              # print(type(scale), scale.shape)
97      return scale.to_numpy()
98
99  # Logistic regression using methods defined in sklearn library
100 def LogisticReg_sklearn(X_train, X_test, y_train, y_test):
101     classifier = LogisticRegression(solver='lbfgs', random_state=0, max_iter=10000)
102     classifier.fit(X_train, y_train)
103     y_pred = classifier.predict(X_test)
104     return y_pred
105
106
107 if __name__ == "__main__":
108     colnames = ['calorific_value', 'nitrogen', 'turbidity', 'style', 'alcohol', 'sugars'
        , 'bitterness', 'beer_id',
109                 'colour', 'degree_of_fermentation']
110     df1 = pd.read_csv(r'beer.txt', sep="\t", header=None, names=colnames)
111     new_cols = ['calorific_value', 'nitrogen', 'turbidity', 'alcohol', 'sugars',
        'bitterness', 'beer_id', 'colour',
112                 'degree_of_fermentation']
113
114     X = pd.DataFrame(df1.loc[:, new_cols])
115     X2 = Scalar_function(X)
116
117     y_data = df1.iloc[:, 3]
118     sys.stdout = open("Logistic_Algo_Accuracy.txt", "w")
119     print("Generating report for accuracy of multiple iterations of Logistic algorithm
        with different samples.")
120     for i in range(1, 11):
121         print("\n ------------------Iteration # ", i, "------------------")
122         X_train, X_test, y_train, y_test = train_test_split(X2, y_data, test_size=0.33)
123         obj1 = Logistic_Multilinear(n_iteration=10000).train_algo(X_train, y_train)
124
```

```
125             Y_predicted = obj1.test_algo(X_test)
126             DF2 = pd.DataFrame({"y_predicted": Y_predicted, "y_test": y_test})
127
128
129             csv_name = "Predicted_MyAlgo.csv"
130
131             DF2.to_csv(csv_name, mode='a', header=True)
132             score1 = round(obj1.accuracy_functn(Y_predicted, y_test) * 100, 2)
133             print("The accuracy of the developed model is ", score1, "%")
134
135             y_pred = LogisticReg_sklearn(X_train, X_test, y_train, y_test)
136             print("Accuracy of Logistic Regression using sklearn is : ", round(
                accuracy_score(y_test, y_pred) * 100, 2),
137                   "%")
138
139             DF3 = pd.DataFrame({"y_predicted": y_pred, "y_test": y_test})
140             csv_name1 = "Predicted_SKLearnAlgo.csv"
141             DF3.to_csv(csv_name1, mode='a', header=True)
142         obj1.plot_cost(obj1.cost)
143         sys.stdout.close()
144
```