

Fundamentals of Machine Learning

Lab Assignment-5

Name:Hitha Choudhary G
USN:22BTRAD015
Branch:CSE in AI&DE

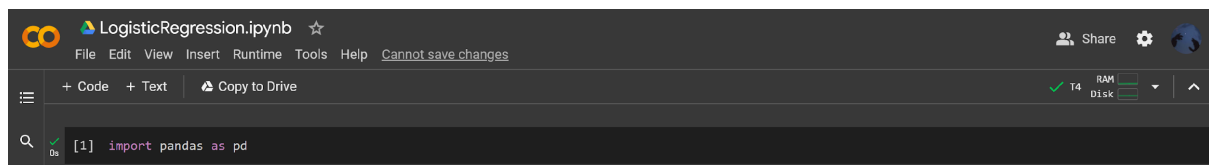
Implementing LogisticRegression algorithm.

One of the most widely used machine learning algorithms, under the category of supervised learning, is logistic regression. With a given collection of independent factors, it is used to predict the categorical dependent variable.

With logistic regression, the result of a categorical dependent variable is predicted. As a result, a discrete or category value must be the result. Instead of providing the exact values, which are 0 and 1, it provides the probabilistic values, which fall between 0 and 1. It can be either Yes or No, 0 or 1, true or False, etc.

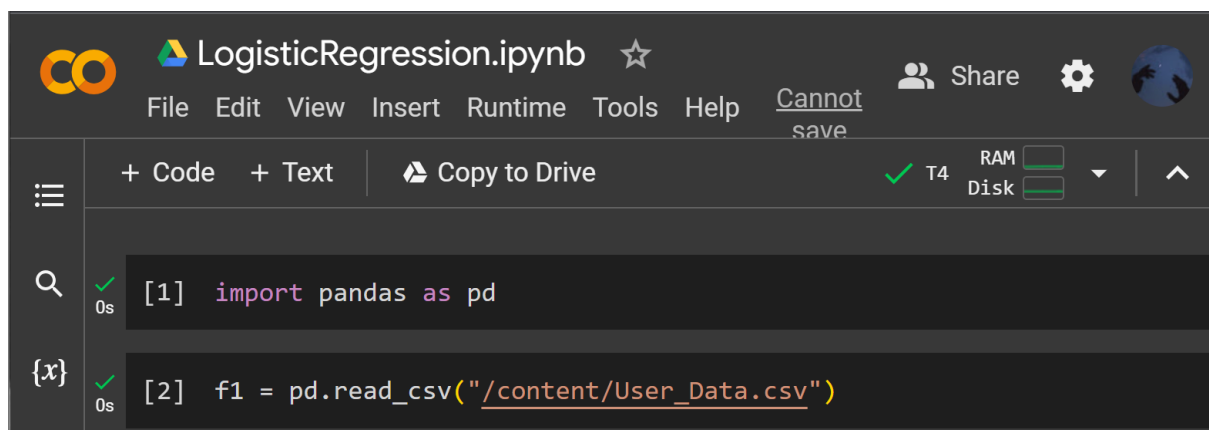
Let's look into the implementation of logistic regression in python.

Importing the pandas library



The screenshot shows a Jupyter Notebook titled "LogisticRegression.ipynb". The interface includes a top bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. Below the menu bar, there are buttons for "+ Code", "+ Text", and "Copy to Drive". The first code cell is visible, containing the code `[1] import pandas as pd`. The cell is marked with a green checkmark and "0s" indicating successful execution.

Importing the dataset "user_data"



The screenshot shows the same Jupyter Notebook interface with two code cells. The first cell contains `[1] import pandas as pd`. The second cell contains `[2] f1 = pd.read_csv("/content/User_Data.csv")`. Both cells are marked with green checkmarks and "0s", indicating successful execution. The notebook title "LogisticRegression.ipynb" and the "Cannot save" status are visible at the top.

Now we run the code to print only the first five rows present in the dataset.

```
[6] f1 = pd.read_csv("/content/User_Data.csv")

f1.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

By executing the below code we can see that the “gender” column is getting distinguished as “Gender_Male” and “Gender_Female”.

This is done by “get_dummies” keyword in python where it converts the categorical values into dummy/indicator values. Each variable is converted as 0/1 value.

```
[8] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[10] f2 = pd.get_dummies(f1, columns=["Gender"])

f2.head()
```

	User ID	Age	EstimatedSalary	Purchased	Gender_Female	Gender_Male
0	15624510	19	19000	0	0	1
1	15810944	35	20000	0	0	1
2	15668575	26	43000	0	1	0
3	15603246	27	57000	0	1	0
4	15804002	19	76000	0	0	1

Importing the required ML libraries

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Now, we will extract the dependent and independent variables from the given dataset.

In the below code we have taken [1,2,4,5] for x because our independent variables are present in those columns and [3] for y because our dependent variable is present column 3.

```
X_data = f2.iloc[:, [1,2,4,5]].values
Y_data = f1.iloc[:, 3].values
```

Now we split the data into training set and test set. The code is given below.

```
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size = 0.2)
```

We will do feature scaling to get accurate result of predictions.

```
#feature scaling
ss = StandardScaler()
x_train_ss = ss.fit_transform(X_train)
x_test_ss = ss.fit_transform(X_test)
```

The function counts the items from given array and gives output in the form of a number as size.

```
[16] X_train, X_test, Y_train, Y_test = train_test_split(f1[["Age", "EstimatedSalary"]], f1.Purchased, test_size = 0.2)

[17] X_train.size
```

Fitting Logistic Regression to the Training set: We will train the dataset using the training set.

```
[18] model = LogisticRegression()
      #model.fit(X_train, Y_train)
      model.fit(x_train_ss, Y_train)
```

Predicting the test result: After training our model on the training set, we will now predict the result using test set data. The below function returns the predicted output values as a one dimensional array.

```
model.predict(x_test_ss)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

We now utilize the `model.predict()` function, which accepts `x_test_ss` as a parameter, to forecast the testing data.

Using the knowledge from the training data, the `model.predict()` function will forecast the values of the testing data's output.

Logistic regression uses the tan function, which has a result of either 1 or 0, which indicates either yes or no (or true or false), model.predict() can only

return a value of 1 or 0. You can check its performance with “.predict_proba()”, which returns the matrix of probabilities that the predicted output is equal to zero or one

```
model.predict_proba(x_test_ss)

array([[0.647973, 0.352027],
       [0.58814962, 0.41185038],
       [0.56199939, 0.43800061],
       [0.68999273, 0.31000727],
       [0.53033078, 0.46966922],
       [0.71221441, 0.28778559],
       [0.52257532, 0.47742468],
       [0.68771557, 0.31228443],
       [0.64100118, 0.35899882],
       [0.71445897, 0.28554103],
       [0.64559131, 0.35440869],
       [0.6710304, 0.3289696],
       [0.76707056, 0.23292944],
       [0.67207395, 0.32792605],
       [0.60411411, 0.39588589],
       [0.69758494, 0.30249506],
       [0.64745865, 0.35254135],
       [0.7553771, 0.2446229],
       [0.53814177, 0.46185823],
       [0.67230153, 0.32769847],
       [0.55869083, 0.44130917],
```

Output

```
[0.62416559, 0.37583441],
[0.507332589, 0.49267411],
[0.57967597, 0.42032403],
[0.61213767, 0.38786233],
[0.547332463, 0.45267537],
[0.64414745, 0.35585255],
[0.5634495, 0.4365505],
[0.6253573, 0.3746427],
[0.60613402, 0.39386598],
[0.57514736, 0.42485264],
[0.51722187, 0.48277813],
[0.6224623, 0.3775377],
[0.60664119, 0.39335881],
[0.62879985, 0.37120015],
[0.66675084, 0.33324916],
[0.52505563, 0.47494437],
[0.65034436, 0.34965564],
[0.62513169, 0.37486831],
[0.53773328, 0.46226672],
[0.71798073, 0.28201927],
[0.5752447, 0.4247553],
[0.66138485, 0.33861515],
[0.64419611, 0.35580389],
[0.61888304, 0.38111696],
[0.63564035, 0.36435965],
[0.67542086, 0.32457914],
[0.72016716, 0.27983284],
[0.5868381, 0.4131619],
[0.57616518, 0.42383482],
[0.67757886, 0.32242114],
[0.68273502, 0.31726498],
[0.73770759, 0.26229241],
[0.71227225, 0.28772775],
[0.68615547, 0.31384453],
[0.68781895, 0.31218105],
[0.74998168, 0.25001832],
[0.68390609, 0.31609391],
[0.68802566, 0.31197434],
[0.70711813, 0.29288187],
```

```
✓ [20] [0.57616518, 0.42383482],
0s [0.67757886, 0.32242114],
[0.68273502, 0.31726498],
[0.73770759, 0.26229241],
[0.71227225, 0.28772775],
[0.68615547, 0.31384453],
[0.68781895, 0.31218105],
[0.74998168, 0.25001832],
[0.68390609, 0.31609391],
[0.68802566, 0.31197434],
[0.70711813, 0.29288187],
[0.6778174 , 0.3221826 ],
[0.68341859, 0.31658141],
[0.68873075, 0.31126925],
[0.68450229, 0.31549771],
[0.71552119, 0.28447881],
[0.64229134, 0.35770866],
[0.68230773, 0.31769227],
[0.6523943 , 0.3476057 ],
[0.61304295, 0.38695705],
[0.66617148, 0.33382852],
[0.73046217, 0.26953783],
[0.62392885, 0.37607115],
[0.5779074 , 0.4220926 ],
[0.74207484, 0.25792516],
[0.5493803 , 0.4506197 ],
[0.53280682, 0.46719318],
[0.60516659, 0.39483341],
[0.61128993, 0.38871007],
[0.72174016, 0.27825984],
[0.603721 , 0.396279 ]]
```

Test the accuracy of the result: We are using “.score” to get the accuracy of our model.

```
model.score(x_test_ss, Y_test)
0.65
```

By observing the above output we can say that the result is the ratio of the number of correct predictions to the number of observations. The accuracy of our model is 65%.

Github Link: <https://github.com/hithachoudhary/machinelearning>