به نام خدا

تمرین دوم مباحث ویژه

ساخت یک کلاسیفایر با استفاده از الگوریتم رگرسیون لاجستیک و فرضیه خط

استاد دولتشاهی

پریسا مبارک

40211415006

شرح تمرین:

یک فایل اکسل داریم. میخواهیم از طریق روش رگرسیون لاجستیک چندمتغیره با استفاده از رابطه نرمال و گرادیان کاهشی، یک مدل آموزش دهیم.درواقع هدف این هست که از طریق رگرسیون لاجستیک، یک کلاسیفایر انجام دهیم. در ادامه نیز همانند تمرین قبل آن را predict کنیم یعنی در نهایت یک مجموعه داده unseen به مدل بدهیم و از این طریق میزان کارایی مدلی که آموزش دادیم را بسنجیم.

برای این کار تمام ستون ها به جز ستو آخر را به عنوان ورودی و ستون آخر را به عنوان خروجی و هدف در نظر میگیریم. هم چنین 70 درصد داده ها را به داده های آموزشی و 30 درصد آن را نیز به مجموعه test اختصاص میدهیم.

این تمرین دقیقا همانند تمرین قبلی اما با کمی تغییر انجام میشود. دو که باید ایجاد کنیم در توابع فرضیه و خطا می باشد. اما قبل از آن چون رگرسیون لجستیک یه روش دسته بندی میباشد، و در این تمرین یک رگرسیون لجستیک باینری داریم، مقادیر خروجی را به صفر و یک تقسیم میکنیم:

```
x, y = df[:, :-1], df[:, -1]

# تبدیل Y (صفر و یک) به برچسب‌های دودویی
y_binary = np.where(y >= np.median(y), 1, 0)


x_train, x_test, y_train, y_test = train_test_split(x, y_binary, test_size=0.3, random_state=42)
```

تابع فرضیه را به این صورت تغییر میدهیم:

```
def h(x, theta):   #تغییر تابع فرضیه
    z = x.dot(theta)
    return 1 / (1 + np.exp(-np.clip(z, -500, 500)))
```

در کد بالا تابع سیگموید را روی تابع فرضیه تمرین قبل، اعمال میکنیم. به این صورت که z را برابر با تابع فرضیه تمرین قبل گذاشته و در خط بعد تابع سیگموید را روی آن اعمال کردم. همچنین به علت رفع خطای overflow از دستورclip برای محدود کردن مقادیر z به محدوده 500 تا 500- استفاده میکنیم.

(این خطا در هنگام استفاده از توابع لگاریتمی ممکن هست به وجود بیاد و ما چون در ادامه از توابع لگاریتمی استفاده کردیم و با خطا مواجه شدیم، در اینجا z را محدود کردم.)

تغییر دوم نیز مربوط به تابع خطا میباشد. به جای mse از تابع زیر استفاده میکنیم:

```
def log_loss(y_pred, y_true):  #تغییر تابع خطا
    epsilon = 1e-10  # برای جلوگیری از صفر شدن لگاریتم صفر
    loss = -y_true * np.log(y_pred + epsilon) - (1 - y_true) * np.log(1 - y_pred + epsilon)
    return loss.mean()
```

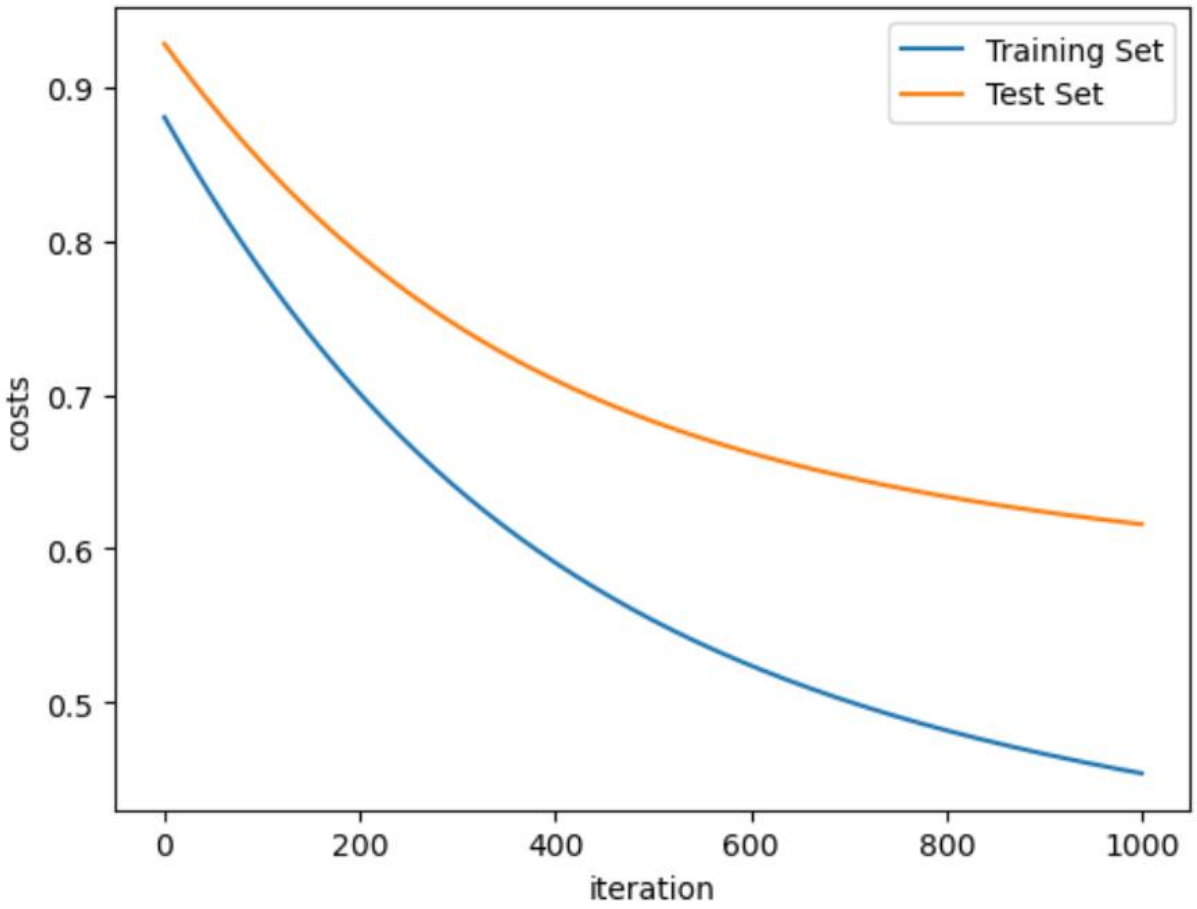
بقیه مراحل مشابه تمرین قبل می باشد. همچنین میتوانیم در آخر دقت را نیز حساب کنیم.برای این کار همانند تصویر زیر یک تابع نوشتم و در نهایت دقت را چاپ میکنیم:

```
# تابع محاسبه دقت
def accuracy(theta, x, y):
    y_pred = h(x, theta)
    y_pred_class=np.where(y_pred >= 0.5, 1, 0)
    acc = np.mean(y_pred_class == y) * 100
    return acc

# محاسبه دقت برای مجموعه‌های آموزش و آزمون
train_accuracy = accuracy(theta, xn_train, y_train)
test_accuracy = accuracy(theta, x_test_n, y_test)

print(" train_accuracy: {:.2f}%".format(train_accuracy))
print(" test_accuracy: {:.2f}%".format(test_accuracy))
```

خروجی های زیر به ترتیب نمودار خطا برای مجموعه آموزش و تست و سپس نمایش دقت نهایی آموزش و تست میباشند :




```
train_accuracy: 89.27%
test_accuracy: 84.27%
```