HW2

LOGISTIC REGRESSION

Στην λογιστική παλινδρόμηση χρησιμοποιήσαμε τα δεδομένα 'PU'.Ο κώδικας είναι γραμμένος σε Python.

Το πρόγραμμα αρχικά διαβάζει όλα τα ονόματα όλων των αρχείων που βρίσκονται εσωτερικά του root φακέλου "pu_corporpora_public" και τα τοποθετεί σε μία λίστα αρχείων προς ανάγνωση.

```
17  # Read files from every folder and return a list containing them
18  def readFiles(root):
19      files = [] # List of files to be returned
20      for r, d, f in os.walk(root):# r=root, d=directories, f = files
21          for file in f:
22              if '.txt' in file:
23                  files.append(os.path.join(r, file))
24      return files
25
```

Διαβάζοντας τα δεδομένα από κάθε αρχείο , δημιουργείται μία λίστα Μηνυμάτων τα οποία έχουν τρία χαρακτηριστικά , Subject , Body και Spam(0-1).Το Subject και το Body είναι περασμένα λίστες λέξεων και το Spam Boolean.

```
26  # Input : a list of filenames
27    # Output : a list of Message objects contained within these files
28  def parseMails(files):
29      Messages = [] # Messages list to be returned
30      for f in files:
31          fname = f.split('/')[-1]
32          with open(f,"r") as fopen:
33              Subject = fopen.readline().rstrip().split(' ')[1:]
34              Body = []
35              lamb = (lambda x:not "legit" in x)
36              Spam = lamb(fname)
37              for line in fopen:
38                  for word in line.rstrip().split(' '):
39                      Body.append(word)
40              Messages.append(Message(Subject ,Body ,Spam ))
41
42      return Messages
```

Έχοντας τα δεδομένα σε κατάλληλη μορφή , μπορούμε και δημιουργούμε λεξικά διατρέχοντας τα Messages. Διαχωρίσαμε τα λεξικά σε λέξεις που εμφανίζονται στο Subject/Body και για Spam/Ham , έτσι δημιουργήσαμε 4 λεξικά. Θεωρώντας ότι ανάλογα με το εάν μία λέξη εμφανίζεται στο Subject ή στο Body , έχει διαφορετική πιθανότητα το μήνυμα να ανήκει σε Spam η Ham , η ακρίβεια των αποτελεσμάτων μας αυξήθηκε περίπου 5 %.(τα λεξικά είναι σε μορφή Counter της Python).

```
# Loop through every message and fill the dictionary
for m in Messages:
    if m.Spam is True:
        for word in set(m.Subject):
            spam_dict[word] += 1
    else:
        for word in set(m.Subject):
            ham_dict[word] += 1
```

Τα λεξικά αυτά θα χρησιμοποιηθούν στην συνέχεια για να επιλεχθούν οι κατάλληλες λέξεις τις οποίες θα τοποθετήσουμε και στο διάνυσμα ιδιοτήτων.

Δημιουργούμε ένα νέο λεξικό με τις ίδιες λέξεις του spam_dict, το οποίο όμως αντί για το πλήθος εμφάνισης της λέξης , έχει την τιμή z = πόσες φορές εμφανίζεται σε σπαμ / πόσες φορές εμφανίζεται συνολικά η λέξη. Ταυτόχρονα 'πετάμε' τις λέξεις που εμφανίζονται λιγότερες από χ φορές.

```
# Loop through every known word and fill out flospam dict
for word in spam_dict:
    hc = ham_dict[word]
    sc = spam_dict[word]
    if hc == sc     : x = 0
    elif hc == 0    : x = 1
    elif sc == 0    : x = -1
    else:
        if hc > sc:
            x = - sc / (hc+sc)
        else:
            #print (sc ,'/' , hc+sc )
            x = sc / (hc+sc)
            #print (x )
    if sc > 3:
        flospam[word] = x
```

ΕΚΠΑΙΔΕΥΣΗ

Έχοντας έτοιμο το λεξικό συνεχίζουμε χωρίζοντας τυχαία τα δεδομένα σε train και test.

```
# Shuffle and split into test and train data
random.shuffle(Messages)
Train_Data = Messages[:6000]
Test_Data = Messages[6000:]
```

Τα πρώτα 6000 mail κρατάμε για την εκπαίδευση , ενώ τα υπόλοιπα για test.

Αρχικοποιούμε ένα διάνυσμα μήκους όσο και το πλήθος των ιδιοτήτων που έχουμε .Επιλέξαμε να χρησιμοποιήσουμε 102 ιδιότητες, οι 2 πρώτες παράγονται από το άρθροισμα του flospam , και οι υπόλοιπες είναι οι top λέξεις από το flospam.

```
# Make a weight list size N , value x
def init_Weights(N ,x):
    Weights = []
    for i in range(N+1):
        Weights.append(x)
    return Weights
```

Σε κάθε παράδειγμα εκπαίδευσης παράγουμε το διάνυσμα ιδιοτήτων και καλούμε την μέθοδο ανανέωσης των βαρών .

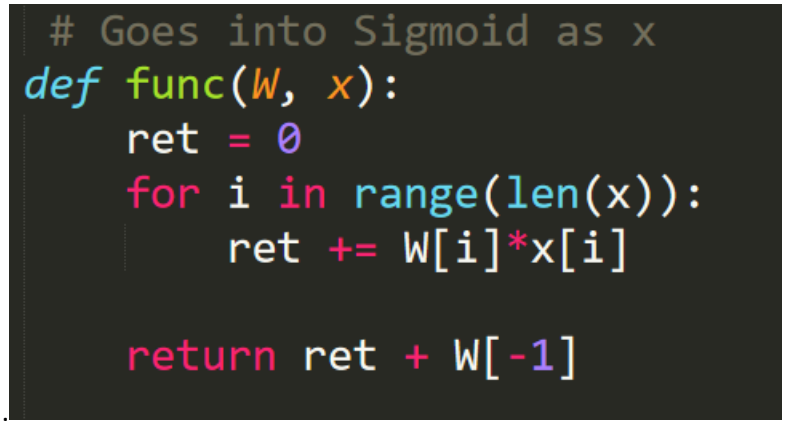
```
# Update Weights
Weights = update_weights(X ,y ,Weights ,res,0.01)
```

```
# Weight update formula
 # b = b + alpha * (y - prediction) * prediction * (1 - prediction) * x
def update_weights(X ,y ,Weights ,pred, a):
    mul = a*(y - pred)*pred*(1- pred)
    for i in range(len(Weights)-1):
        w = Weights[i]
        Weights[i] = abs( w + mul*X[i])
    Weights[-1] =  a*(y - pred)*pred*(1- pred)
    return Weights
```

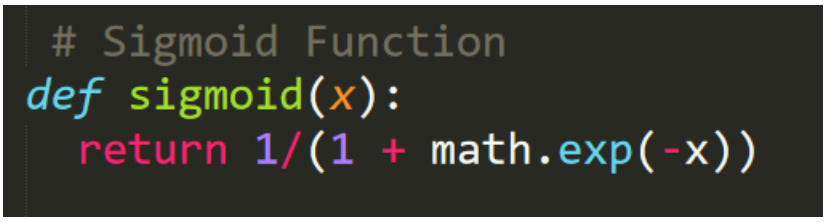
Στο τέλος της εκπαίδευσης επιστρέφεται το διάνυσμα βάρους στην main.

TESTING

Παρόμοια με την εκπαίδευση , κατά την διάρκεια του test παράγεται ένα διάνυσμα ιδιοτήτων του παραδείγματος .Αυτήν την φορά όμως θα χρησιμοποιήσουμε τα βάρη που μάθαμε , και θα δώσουμε το διάνυσμα βάρους και ιδιοτήτων στην συνάρτηση func



Η func με την σειρά της , επιστρέφει το αποτέλεσμα της στην sigmoid , η οποία θα μας δώσει και την εκτίμηση για το εάν το παράδειγμα μας είναι spam η ham.



ΑΠΟΤΕΛΕΣΜΑΤΑ

Δοκιμάσαμε επίσης την βιβλιοθήκη scikit και εκπαιδεύσαμε ένα μοντέλο χρησιμοποιώντας τις έτοιμες βιβλιοθήκες Λογιστικής Παλινδρόμησης που παρέχει. Ο κώδικας βρίσκεται στο φάκελο.

Το πρόγραμμα μας πιάνει μέσο όρο 94 % ακρίβεια σε δεδομένα τεστ. Με το scikit καταφέραμε 90%.

Διάγραμμα ακρίβειας του αλγόριθμου εκπαίδευσης μας