Αναγνώριση Προτύπων

Εργασία 1

2020-2021

5ο εξάμξηνο

Παραδοτέο από

Στέφανος Μάριος Δανιήλ p16026

Αθανάσιος Δασούλας p18033

Χρήστος Κατέβας p18068

Επιβλέπων καθηγητής: Διονύσης Σωτηρόπουλος

Βιβλιοθήκη Pandas

Εισάγονται τα δεδομένα στο πρόγραμμα

```
connection = sqlite3.connect('database.sqlite')

match = pd.read_sql("select id, home_team_goal, away_team_goal, B365H, B365D, "
                    "B365A, BWH, BWD, BWA, IWH, IWD, IWA, LBH, LBD, LBA from Match", connection, index_col="id").dropna(axis='rows')
attributes = pd.read_sql("Select id, team_fifa_api_id, team_api_id, date, buildUpPlaySpeed, buildUpPlaySpeedClass, buildUpPlayDribbling,
buildUpPlayDribblingClass, buildUpPlayPassing, buildUpPlayPassingClass, chanceCreationPassing, chanceCreationPassingClass,
chanceCreationCrossing, chanceCreationCrossingClass, chanceCreationShooting , chanceCreationShootingClass ,
chanceCreationPositioningClass , defencePressure , defencePressureClass , defenceAggression, defenceAggressionClass , defenceTeamWidth,
defenceTeamWidthClass , defenceDefenderLineClass from Team_Attributes",connection,index_col="id").dropna(axis='rows')

```

Τα αποτελέσματα καταχωρούνται στους πίνακες(επέκταση πινάκων) ως εξής:

Εάν η γηπεδούχος ομάδα κερδίσει τον αγώνα, καταχωρείται στον πίνακα η τιμή ‘1’, εάν χάσει των αγώνα καταχωρείται η τιμή ‘-1’. Σε περίπτωση ισοπαλίας, το ‘0’

```

for i in range(len(Home_goal)):
    if Home_goal[i]>Guest_goal[i]:
        Results.append(1)
    elif Home_goal[i]<Guest_goal[i]:
        Results.append(0)
    else:
        Results.append(-1)

```

Ο Αλγόριθμος “LMS”

Η συνάρτηση LMS() υπολογίζει και επιστρέφει την τιμή w, που περιέχει το αποτέλεσμα.

(Όσο αυξάνουμε το range, τόσο αυξάνεται η ακρίβεια)

```
def LMS(x,y):
    x = np.hstack((np.ones((len(x), 1)), x))
    w = np.random.rand(x.shape[1], 3)
    out , out[np.arange(len(y)), y] = np.zeros((len(y), 3)),1

    for t in range(len(out)):
        i = int(Results[t])
        out[t,i] = 1
    loss = []
    for i in range(100): #increase the value and get more accurate results (high temperatures)
        predictions = x.dot(w)

        diffs = predictions - out
        l = np.mean(np.square(diffs))
        loss.append(l)
        gradent = x.T @ diffs
        w -= 0.000001*gradent
    return w

```

Ο αλγόριθμος μέσου τετραγωνικού σφάλματος βοηθά στη κατανόηση της θεωρίας των γραμμικών προσαρμοζόμενων φίλτρων που χρησιμοποιούν απλό γραμμικό νευρώνα. Το ζητούμενο είναι να υπολογιστούν οι βέλτιστες τιμές του w, έτσι ώστε να ελαχιστοποιείται κάποιο κατάλληλο συνολικό σφάλμα, J. Αυτό είναι συνήθως μια συνάρτηση του σφάλματος e. Στην προκειμένη ο ταξινομητής υλοποιεί την συνάρτηση g*k (ψ*κ (m)): R^3→{H,D,A} Στον κώδικα έχει δημιουργηθεί η συνάρτηση LMS() όπου υπολογίζει την βέλτιση τιμή του w.

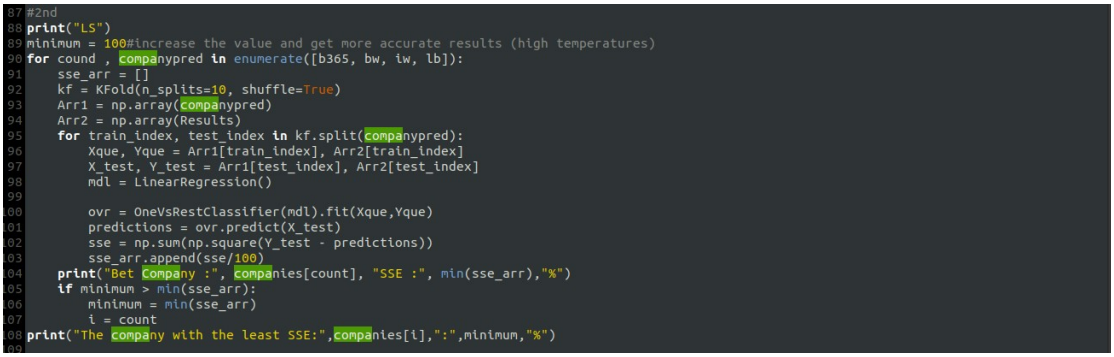
```
print("LMS")
max = -1
for count , companypred in enumerate([b365, bw, iw, lb]):
    kf = KFold(n_splits=10, shuffle = True)
    Acc = []
    Arr1 = np.array(companypred)
    Arr2 = np.array(Results)
    for train_index, test_index in kf.split(companypred):
        Xque , Yque = Arr1[train_index], Arr2[train_index]
        X_test, Y_test = Arr1[test_index], Arr2[test_index]
        res = LMS(Xque,Yque)
        test_bias = np.hstack((np.ones((len(X_test),1)),X_test))
        estimates = np.dot(test_bias,res).argmax(axis=1)
        accuracy = np.mean(estimates==Y_test)
        Acc.append(accuracy)
    print("Bet Company :", companies[count], "Accuracy results:", np.mean(Acc))
    if(max < np.mean(Acc)):
        max = np.mean(Acc)
        i= count

print("Best company : ",companies[i],"with (result)",max)
```

Για κάθε εταιρία πραγματοποιούμε 10-fold cross validation. Το μοντέλο εκπαιδεύται. Προσθέτονται Bias. Και πραγματοποιούνται οι προβλέψεις. Συγκρίνονται τα αποτελέσματα των προβλέψεων με των Y(Y_test) και ταξινομούνται οι καλύτερες προγνώσεις. Για τις ταξινομήσεις χρησιμοποιείται η βιβλιοθήκη kfold και numpy.

Ερώτημα 2

Η μέθοδος των λιγότερων τετραγώνων αφορά την εκτίμηση παραμέτρων ελαχιστοποιώντας τις τετραγωνικές αποκλίσεις μεταξύ των παρατηρούμενων δεδομένων και των αναμενόμενων. Στην προκειμένη ο ταξινομητής υλοποιεί την συνάρτηση g*k (ψ*κ (m)): R^3→{H,D,A}.



```
#2nd
print("LS")
minimum = 100#increase the value and get more accurate results (high temperatures)
for cound , companypred in enumerate([b365, bw, iw, lb]):
    sse_arr = []
    kf = KFold(n_splits=10, shuffle=True)
    Arr1 = np.array(companypred)
    Arr2 = np.array(Results)
    for train_index, test_index in kf.split(companypred):
        Xque, Yque = Arr1[train_index], Arr2[train_index]
        X_test, Y_test = Arr1[test_index], Arr2[test_index]
        mdl = LinearRegression()

        ovr = OneVsRestClassifier(mdl).fit(Xque,Yque)
        predictions = ovr.predict(X_test)
        sse = np.sum(np.square(Y_test - predictions))
        sse_arr.append(sse/100)
    print("Bet Company :", companies[count], "SSE :", min(sse_arr),"%")
    if minimum > min(sse_arr):
        minimum = min(sse_arr)
        i = count
print("The company with the least SSE:",companies[i],":",minimum,"%")
```

Όπως και στο 1ο ερώτημα

Για κάθε στοιχηματική εταιρία πραγματοποιούμε 10-fold cross validation. Το μοντέλο εκπαιδεύται. Προσθέτονται Bias. Και πραγματοποιούνται οι προβλέψεις. Συγκρίνονται τα αποτελέσματα των προβλέψεων με των Y(Y_test) και ταξινομούνται οι καλύτερες προγνώσεις. Για τις ταξινομήσεις χρησιμοποιείται η βιβλιοθήκη kfold και numpy.

Τα μοντέλα εκπαιδεύουμε τα μοντέλα, σορτάρονται από Linear Regression και OneVsRestClassifier. Υπολογίζονται οι προβλέψεις και βγαίνει το Sum Square Error για κάθε μία από τις στοιχηματικές εταιρείς. Τέλος υπολογίζεται η μικρότερη τιμή.

Συγκρίνουμε τις ακρίβειες , και εμφανίζεται η εταιρεια με την μεγαλύτερη ακρίβεια.

Βιβλιογραφία

1) https://github.com/PennyNeko/ClusteringAlgorithms

2) https://github.com/ThomIves/LeastSquaresPurePy

3) https://matousc89.github.io/padasip/sources/filters/lms.html

4) https://en.wikipedia.org/wiki/Least_mean_squares_filter#:~:text=Least%20mean%20squares%20(LMS)%20algorithms,desired%20and%20the%20actual%20signal)

5) https://en.wikipedia.org/wiki/Least_squares