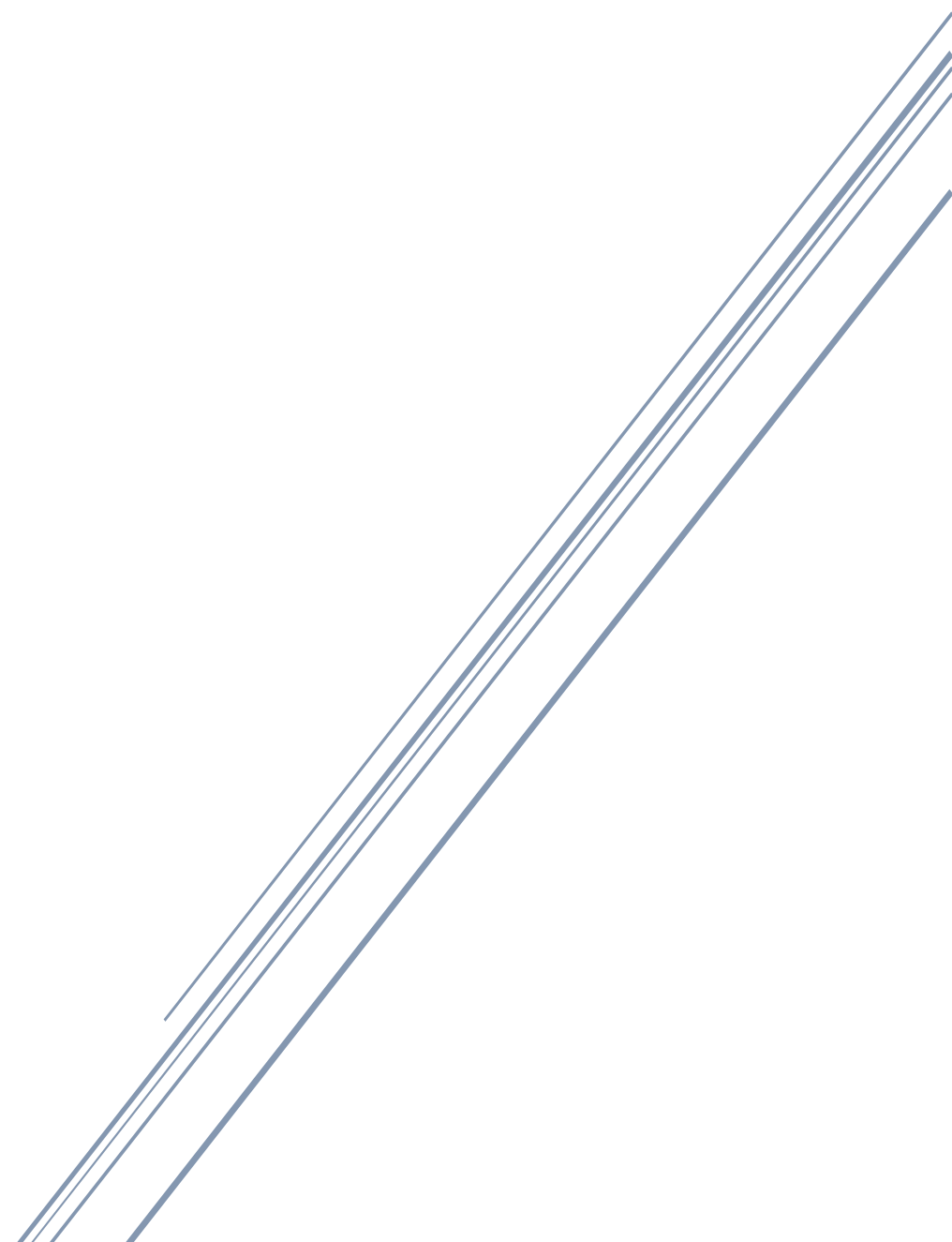


Αναγνώριση Προτύπων

Απαλλακτική Εργασία

Ακαδημαϊκό Έτος 2020 - 2021



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

Δημήτρης Ματσαγγάνης, Π17068

Περιεχόμενα

Εισαγωγή Εργασίας.....	3
Κεντρική Ιδέα Υλοποίησης.....	5
Μορφοποίηση Παρεχόμενης Βάσης Δεδομένων	6
Θέμα 1: Αλγόριθμος Ελάχιστου Μέσου Τετραγωνικού Σφάλματος - Least Mean Squares	12
Απεικόνιση Αποτελεσμάτων – Γραφήματα	20
Θέμα 2: Αλγόριθμος Ελάχιστου Τετραγωνικού Σφάλματος - Least Squares	22
Απεικόνιση Αποτελεσμάτων – Γραφήματα	29
Θέμα 3: Πολυστρωματικό Νευρωνικό Δίκτυο – MultiLayer Neural Network .	31
Απεικόνιση Αποτελεσμάτων – Γραφήματα	37
Κεντρικό Αρχείο Εκτέλεσης – main	40
K-Fold Cross Validation Algorithm	43
Οδηγίες Εκτέλεσης Προγράμματος	48
Βιβλιογραφία.....	49
Περιεχόμενα Απεσταλμένου Αρχείου	51

Εισαγωγή Εργασίας

Απαλλακτική Εργασία 2020

Θέμα : Πρόγνωση Αποτελεσμάτων Ποδοσφαιρικών Αγώνων με χρήση Αλγορίθμων Μηχανικής Μάθησης

Στόχος της συγκεκριμένης εργασίας είναι η ανάπτυξη αλγορίθμων μηχανικής μάθησης για την πρόβλεψη του αποτελέσματος ενός ποδοσφαιρικού αγώνα. Το σύνολο των δεδομένων που θα χρησιμοποιήσετε βρίσκεται στην παρακάτω δικτυακή τοποθεσία, <https://www.kaggle.com/hugomathien/soccer>, υπό την μορφή μιας βάσης δεδομένων SQLite.

Μάλιστα, η ωφέλιμη πληροφορία για την υλοποίηση των ζητούμενων μηχανισμών μάθησης είναι αποθηκευμένη στους πίνακες **Match** και **Team_Attributes**.

Θεωρώντας το σύνολο των διαφορετικών αγώνων της βάσης ως $M = \{m_1, \dots, m_N\}$ και το αντίστοιχο σύνολο των διαφορετικών ομάδων ως $T = \{t_1, \dots, t_K\}$, τότε η πιο αφηρημένη αναπαράσταση του κάθε αγώνα μπορεί να πραγματοποιηθεί ως μια διατεταγμένη τριάδα της μορφής $m = \langle h, a, r \rangle$, με $h \neq a$, όπου η $h \in T$ είναι η ομάδα που αγωνίζεται εντός έδρας (home team), $a \in T$ είναι η ομάδα που αγωνίζεται εκτός έδρας (away team) και $r \in \{H, D, A\}$ το αποτέλεσμα του αγώνα. Συγκεκριμένα, το **H** (home win) υποδηλώνει νίκη της ομάδας που αγωνίζεται εντός έδρας, το **D** (Draw) υποδηλώνει την ισόπαλη έκβαση του αγώνα και το **A** (away win) υποδηλώνει την νίκη της ομάδας που αγωνίζεται εκτός έδρας. Αν με $G_m(h) \geq 0$ και $G_m(a) \geq 0$ συμβολίσουμε το πλήθος των τερμάτων που επιτυγχάνονται από τις ομάδες εντός και εκτός έδρας κατά την διεξαγωγή του αγώνα m , τότε η μεταβλητή έκβασης του αγώνα r μπορεί να ορισθεί συναρτήσει της διαφοράς των συνολικών τερμάτων $\Delta G_m(h, a) = G_m(h) - G_m(a)$ ως:

$$r = \begin{cases} H, \Delta G_m(h, a) > 0; \\ D, \Delta G_m(h, a) = 0; \\ A, \Delta G_m(h, a) < 0. \end{cases}$$

Η διαδικασία εκπαίδευσης των εμπλεκόμενων ταξινομητών θα πρέπει να βασιστεί σε ένα σύνολο χαρακτηριστικών γνωρισμάτων της κάθε ομάδας καθώς και σε ένα σύνολο προγνωστικών (odds) για την πιθανή έκβαση του κάθε αγώνα από έναν αριθμό στοιχηματικών εταιρειών. Συγκεκριμένα, κάθε ομάδα $t \in T$ είναι συσχετισμένη με ένα διάνυσμα χαρακτηριστικών $\varphi(t) \in \mathbb{R}^8$ τα οποία αντιστοιχούν στις παρακάτω στήλες του πίνακα **Team_Attributes** {**buildUpPlaySpeed**, **buildUpPlayPassing**, **chanceCreationPassing**, **chanceCreationCrossing**, **chanceCreationShooting**, **defencePressure**, **defenceAggregation**, **defenceTeamWidth**}. Επιπλέον, κάθε αγώνας $m \in M$ είναι συσχετισμένος με τέσσερα διανύσματα προγνωστικών $\psi_k(m) \in \mathbb{R}^3$ με $k \in \{B365, BW, IW, LB\}$ τα οποία αντιστοιχούν στις παρακάτω στήλες του πίνακα **Match** {**B365H**, **B365D**, **B365A**, **BWH**, **BWD**, **BWA**, **IWH**, **IWD**, **IWA**, **LBH**, **LBD**, **LBA**}. Δηλαδή, το κάθε διάνυσμα $\psi_k(m) = [d_k^H(m), d_k^D(m), d_k^A(m)]$ συγκεντρώνει τις στοιχηματικές αποδόσεις για κάθε πιθανή έκβαση του αγώνα m για κάθε στοιχηματική εταιρεία $k \in \{B365, BW, IW, LB\}$. Λάβετε υπόψιν πως υπάρχουν εγγραφές στον πίνακα **Match** για τις οποίες τα αντίστοιχα διανύσματα προγνωστικών έχουν μηδενικές τιμές. Οι συγκεκριμένες εγγραφές θα πρέπει να αφαιρεθούν.

Ερωτήματα:

- I. Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Μέσου Τετραγωνικού Σφάλματος (**Least Mean Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφής $g_k(\psi_k(m)) : \mathbb{R}^3 \rightarrow \{H, D, A\}$ για κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.
- II. Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Τετραγωνικού Σφάλματος (**Least Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφής $g_k(\psi_k(m)) : \mathbb{R}^3 \rightarrow \{H, D, A\}$ για κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.
- III. Να υλοποιήσετε ένα πολυστρωματικό νευρωνικό δίκτυο, ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί μια συνάρτηση διάκρισης της μορφής $g(\Phi(m)) : \mathbb{R}^{28} \rightarrow \{H, D, A\}$, όπου το $\Phi(m) \in \mathbb{R}^{28}$ αντιστοιχεί στο πλήρες διάνυσμα χαρακτηριστικών του κάθε αγώνα που δίνεται από την σχέση:
$$\Phi(m) = [\varphi(h), \varphi(a), \psi_{B365}(m), \psi_{BW}(m), \psi_{IW}(m), \psi_{LW}(m)]$$

Παρατηρήσεις:

- I. Για κάθε ταξινομητή που θα υλοποιήσετε θα πρέπει να αναφέρετε την ταξινομητική του ακρίβεια τόσο κατά την φάση της εκπαίδευσης όσο και κατά την φάση του ελέγχου σύμφωνα με την μέθοδο της 10-πλής διεπικύρωσης (**10 fold cross validation**).
- II. Στο αρχείο **EuropeanSoccerDatabaseRetriever.m** σας παρέχετε κώδικας για την άντληση των δεδομένων από την βάση SQLite. Στον φάκελο της εργασίας θα βρείτε και παραδείγματα υλοποίησης πολυστρωματικών δικτύων σε MATLAB.
- III. Παραδοτέα της εργασίας αποτελούν ο **κώδικας** της υλοποίησής σας σε MATLAB ή Python καθώς και ένα συνοδευτικό **κείμενο τεκμηρίωσης**.
- IV. Μπορείτε να εργασθείτε σε ομάδες των **δύο ή τριών ατόμων**.
- V. Καταληκτική ημερομηνία παράδοσης της εργασίας είναι η **τελευταία μέρα** της εξεταστικής περιόδου.

Κεντρική Ιδέα Υλοποίησης

Σκοπός της παρούσας υλοποίησης είναι η ταξινόμηση τριών αποτελεσμάτων - πιθανών ενδεχομένων (νίκη γηπεδούχου ομάδας, ισοπαλία, νίκη φιλοξενούμενης ομάδας) σε πραγματικούς ποδοσφαιρικούς αγώνες.

Τα δεδομένα των αγώνων μας παρέχονται από την [Kaggle](https://www.kaggle.com) μέσω μιας βάσης δεδομένων sqlite (συγκεκριμένα την database.sqlite).

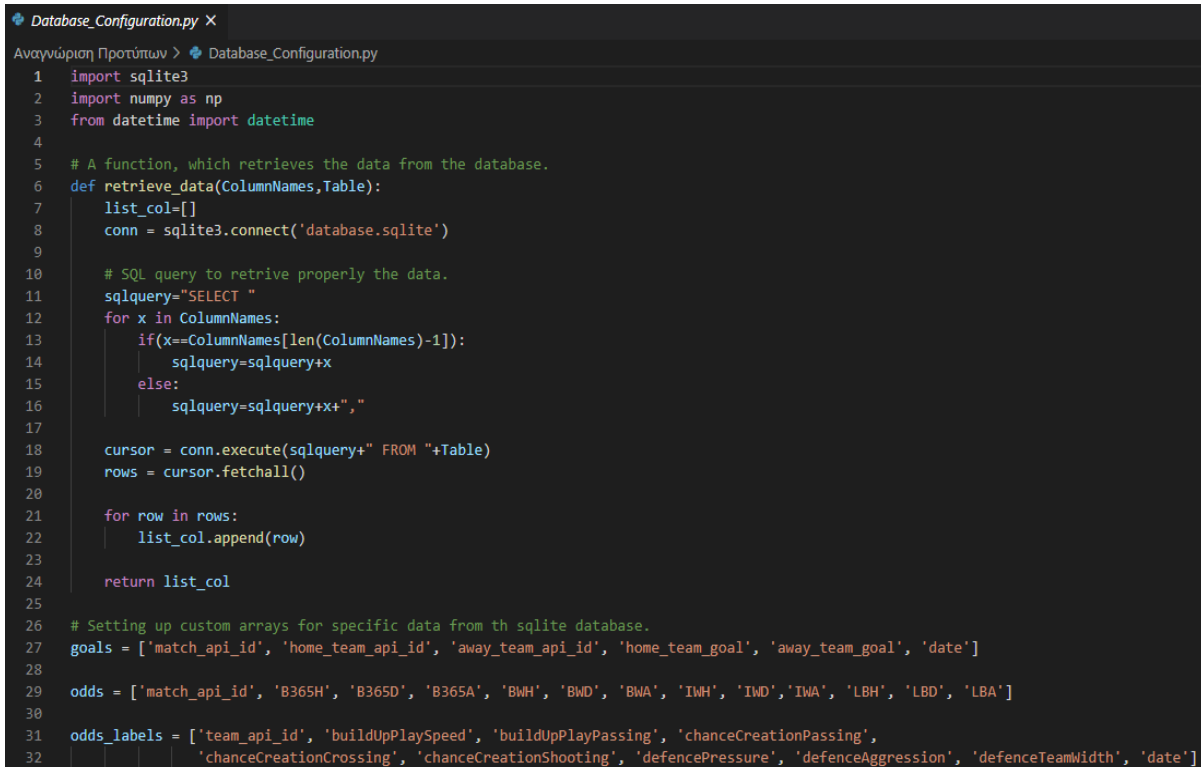
Για την ορθολογική απάντηση των τριών ερωτημάτων της άσκησης θα δημιουργήσουμε τρεις διαφορετικούς ταξινομητές ώστε να συγκρίνουμε τα αποτελέσματά τους, ακολουθώντας τις υποδείξεις της εκφώνησής της άσκησης.

Για τον εκάστοτε ταξινομητή θα υπάρξει ξεχωριστή ενότητα, όπου θα αναλύεται εκτενώς, ενώ θα υπάρχει και ειδική μνεία στο κεντρικό αρχείο ενεργοποίησης που θα δέχεται όρισμα από το χρήστη, το αρχείο τροποποίησης της παρεχόμενης βάσης των δεδομένων (θα κρατάμε μόνο τα ουσιαστικά δεδομένα αναλόγως το ερώτημα) και στη μέθοδο της 10-πλής Διεπικύρωσης (10 Fold Cross Validation).

Μορφοποίηση Παρεχόμενης Βάσης Δεδομένων

Στην ενότητα αυτή θα αναφερθούμε σε ένα από τα κυριότερα σημεία της εργασίας μας, στο αρχείο που θα εξάγει, θα φιλτράρει και θα τροποποιεί τα δεδομένα, τοποθετώντας τα με τον κατάλληλο τρόπο στους πίνακες.

Το αρχείο αυτό ονομάζεται *Database_Configuration.py* και η πρώτη συνάρτηση που χρησιμοποιείται είναι η **retrieve_data**.



```
Database_Configuration.py X
Αναγνώριση Προτύπων > Database_Configuration.py
1 import sqlite3
2 import numpy as np
3 from datetime import datetime
4
5 # A function, which retrieves the data from the database.
6 def retrieve_data(ColumnNames,Table):
7     list_col=[]
8     conn = sqlite3.connect('database.sqlite')
9
10    # SQL query to retrieve properly the data.
11    sqlquery="SELECT "
12    for x in ColumnNames:
13        if(x==ColumnNames[len(ColumnNames)-1]):
14            sqlquery=sqlquery+x
15        else:
16            sqlquery=sqlquery+x+" ,"
17
18    cursor = conn.execute(sqlquery+" FROM "+Table)
19    rows = cursor.fetchall()
20
21    for row in rows:
22        list_col.append(row)
23
24    return list_col
25
26 # Setting up custom arrays for specific data from th sqlite database.
27 goals = ['match_api_id', 'home_team_api_id', 'away_team_api_id', 'home_team_goal', 'away_team_goal', 'date']
28
29 odds = ['match_api_id', 'B365H', 'B365D', 'B365A', 'BWH', 'BWD', 'BWA', 'IWH', 'IWD', 'IWA', 'LBH', 'LBD', 'LBA']
30
31 odds_labels = ['team_api_id', 'buildUpPlaySpeed', 'buildUpPlayPassing', 'chanceCreationPassing',
32               'chanceCreationCrossing', 'chanceCreationShooting', 'defencePressure', 'defenceAggression', 'defenceTeamWidth', 'date']
```

Εικόνα Εφαρμογής 1

Η συγκεκριμένη συνάρτηση αρχικά θα αρχικοποιεί μία κενή λίστα, την τελική λίστα και θα πραγματοποιεί τη σύνδεση με τη βάση δεδομένων – χρησιμοποιώντας τη βιβλιοθήκη *sqlite3* – και στη συνέχεια αφού έχει εξάγει τα δεδομένα της βάσης θα τα τοποθετεί μέσα σε μία λίστα με την χρήση ενός SQL Query.

Στη συνέχεια του αρχείου, ακολουθούν τρεις custom λίστες, οι οποίες χειρίζονται και τροποποιούν τα δεδομένα κατάλληλα για τις συναρτήσεις που θα ακολουθήσουν.

Πιο συγκεκριμένα, η πρώτη custom λίστα, K, που δημιουργείται αφορά τις στοιχηματικές αποδόσεις που δίνονται από τις εκάστοτε εταιρίες .

```
Database_Configuration.py X
Αναγνώριση Προτύπων > Database_Configuration.py
34 # Odds' Table.
35 K = retrieve_data(odds, "Match")
36
37 # Retrieves the odds for 5000 matches (0 to 5000).
38 K = K[:5000]
39
40 # Initialize an empty array, A
41 # to select the odds with 'none' value.
42 A = []
43
44 # An Inner For-loop to retrieve the odds and parse them into
45 # the aforementioned empty array, A.
46 for i in K:
47     for j in i:
48         # An if statement to reject the 'none'
49         # provided odds (A is the no-value odds array).
50         if(j is None):
51             A.append(i)
52             break
53
54 # A For-loop to remove the no-value
55 # odds from the K array
56 for i in A:
57     K.remove(i)
58
59 # Delete A
60 del A
61
62 # Print the K array - testing purposes.
63 # print(K)
64
65 # The table with the Goal Difference, in order
66 # to find out which is the result of the match.
67 G = retrieve_data(goals, "Match")
68
```

Εικόνα Εφαρμογής 2

Προτού παραχθεί ο τελικός πίνακας των αποδόσεων θα απορρίψουμε τις κενές αποδόσεις – τις σειρές (rows) της βάσης που δεν έχουν τιμές. Η παραπάνω διαδικασία γίνεται μέσα από εμφολευμένες δομές επανάληψης – for loops.

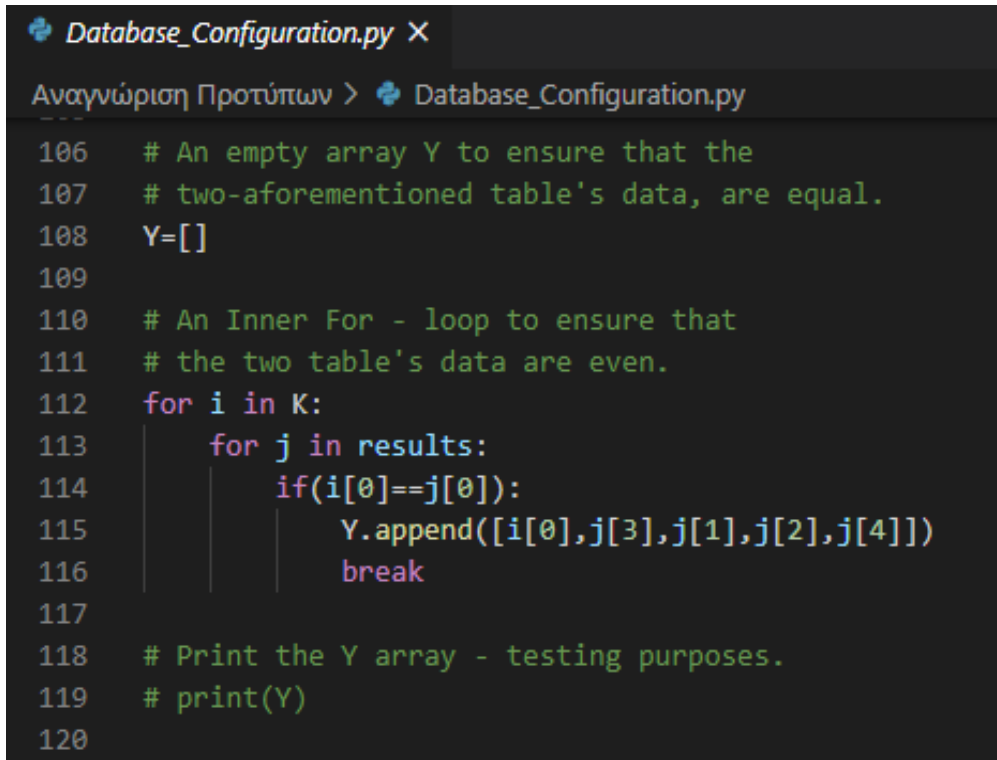
Η δεύτερη εξ αυτών, αφορά τα γκολ που σημειώθηκαν σε κάποιον αγώνα (ο οποίος θα βρίσκεται από το id του, *match_api_id*). Μέσα από αυτή τη διαδικασία βρίσκουμε την διαφορά των γκολ και το τελικό αποτέλεσμα του αγώνα, ανάλογα με τη διαφορά λοιπόν και το πρόσημο αυτής βρίσκουμε το αγώνα.

```
Database_Configuration.py X
Αναγνώριση Προτύπων > Database_Configuration.py
69 # Retrieves the goals for 5000 matches (0 to 5000).
70 G = G[:5000]
71
72 results=[]
73
74 #-----
75 # g[0]: match id.      |
76 # g[1]: home team.    |
77 # g[2]: away team.    |
78 # g[3]: home team's goals. |
79 # g[4]: away team's goals. |
80 # g[5]: match's date.   |
81 #-----
82
83 # A For-loop to get the match's result via Goal Difference.
84 for g in G:
85     # Find Goal Difference as home - away team's goals.
86     GD = g[3] - g[4]
87
88     # If GD > 0, the home team goals are more than
89     # the away's, so home team wins.
90     if(GD > 0):
91         results.append([g[0],g[1],g[2],1,g[5]])
92
93     # Else if GD = 0, the home team goals equals
94     # the away's, so match ends as a draw.
95     elif(GD == 0):
96         results.append([g[0],g[1],g[2],0,g[5]])
97
98     # Else (If GD < 0), the home team goals are
99     # less than the away's, so away team wins.
100    else:
101        results.append([g[0],g[1],g[2],2,g[5]])
102
103    # Print the results array - testing purposes.
104    # print(results)
```

Εικόνα Εφαρμογής 3

Ειδικότερα, εάν η διαφορά των γκολ είναι θετική τότε νικάει η γηπεδούχος ομάδα και λαμβάνει την τιμή “1”, εάν ισούται με το 0 τότε ο αγώνας έληξε ισοπαλία και λαμβάνει την τιμή “0”, ενώ τέλος εάν η διαφορά έχει αρνητικό πρόσημο, τότε νικήτρια ομάδα είναι η φιλοξενούμενη και λαμβάνει την τιμή “2”.

Τέλος, η τρίτη και τελευταία λίστα που δημιουργείται, έχει να κάνει με τις τελικές ετικέτες – labels που θα λαμβάνει κάθε απόδοση.



```
Database_Configuration.py X
Αναγνώριση Προτύπων > Database_Configuration.py
106 # An empty array Y to ensure that the
107 # two-aforementioned table's data, are equal.
108 Y=[]
109
110 # An Inner For - loop to ensure that
111 # the two table's data are even.
112 for i in K:
113     for j in results:
114         if(i[0]==j[0]):
115             Y.append([i[0],j[3],j[1],j[2],j[4]])
116             break
117
118 # Print the Y array - testing purposes.
119 # print(Y)
120
```

Εικόνα Εφαρμογής 4

Συνδυαστικά με τα παραπάνω, η τρίτη λίστα θα επιβεβαιώνει μέσα από μια διαδικασία εμφολευμένων επαναλήψεων, ότι τα δεδομένα βρίσκονται σε πλήρη αντιστοίχιση.

Η δεύτερη συνάρτηση που εμπεριέχεται στο αρχείο *database_configuration.py*, σχετίζεται με το τρίτο ζητούμενο της εργασίας και συγκεκριμένα με την τροποποίηση των δεδομένων του Πολυστρωματικού Νευρωνικού Δικτύου (Multi-Layer Neutral Network).

```
Database_Configuration.py X
Αναγνώριση Προτύπων > Database_Configuration.py
122 # A Function for MLNN (3rd Subject), in order to
123 # concatenate the data from F and K tables, when
124 # the option is selected.
125 def MNN_data():
126
127     # Odds Labels' Table.
128     F = retrieve_data(odds_labels,"Team_Attributes")
129     teams_att = np.array(F)
130     training_data = K
131     target_data = np.array(Y)
132
133     # Get the date.
134     min_dt = datetime.strptime(teams_att[np.argmin(teams_att[:,9])][9], '%Y-%m-%d %H:%M:%S')
135
136     # Console's Message.
137     print('Please wait. Setting up and Fixing some Data!')
138
139     # Initialize array, trained_data_array for the trained data of the MLNN.
140     trained_data_array = []
141
142     # Initialize array, classified_results_array for the classified results
143     # and data to 3 specific classes [0,1,2]=[1,0,0],[0,1,0],[0,0,1].
144     classified_results_array = []
145
146     # Initialize a counter variable.
147     counter = 0
```

Εικόνα Εφαρμογής 5

Η συνάρτηση **MNN_data** καλείται κατά την επιλογή της υλοποίησης για το τρίτο ζητούμενο της άσκησης από την main.

Η συγκεκριμένη συνάρτηση στόχο έχει να κάνει “concatenate” λίστες που αφορούν τις στοιχηματικές αποδόσεις και τα χαρακτηριστικά των ομάδων, ανάλογα με τον εκάστοτε αγώνα.

Ειδικότερα, η σύνδεση των λιστών γίνεται μέσω του κωδικού id του κάθε αγώνα και την ημερομηνία που διεξήχθη ο συγκεκριμένος αγώνας.

Επιπλέον, κρίνεται σκόπιμο να γίνει υλοποίηση της μεθόδου one-hot encode, ώστε να γίνει σωστά η διαδικασία της διακριτοποίησης των κλάσεων και να δημιουργήσουμε ορθολογικά τις κατηγορίες ως εξής:

- κατηγορία 0 [1,0,0]
- κατηγορία 1 [0,1,0]
- κατηγορία 2 [0,0,1]

```
Database_Configuration.py X
Αναγνώριση Προτύπων > Database_Configuration.py
149 # Inner For-loops and if statements, in order to create the final two tables.
150 for i in range(0,len(target_data)):
151     dt2 = datetime.strptime(target_data[i][4], '%Y-%m-%d %H:%M:%S')
152     counter = 0
153
154     for j in range(0,len(teams_att)):
155         if(min_dt.year > dt2.year or counter > 1):
156             break
157
158         # Get the date.
159         dt = datetime.strptime(teams_att[j][9], '%Y-%m-%d %H:%M:%S')
160
161         if((teams_att[j][0] == target_data[i][2] or teams_att[j][0] == target_data[i][3]) and dt2.year == dt.year):
162             counter += 1
163             training_data[i] = np.concatenate([training_data[i], teams_att[j][1:9]], axis=None)
164
165     # Depending the match's result (home, away win or draw), create a new
166     # table - array, in order to classify the data to 3 classes
167     # [0,1,2]=[1,0,0],[0,1,0],[0,0,1] (one-hot encode method).
168     if(len(training_data[i]) > 21):
169
170         # Append data to the trained_data_array.
171         trained_data_array.append(training_data[i][1:])
172
173         # Home team win.
174         if(target_data[i][1] == '1'):
175             classified_results_array.append([1,0,0])
176         # Draw.
177         elif(target_data[i][1] == '0'):
178             classified_results_array.append([0,1,0])
179         # Away team win.
180         else:
181             classified_results_array.append([0,0,1])
182
183     # Returns the tables, in order to get handled from the main.py .
184     return trained_data_array, classified_results_array
```

Εικόνα Εφαρμογής 6

Στο τέλος της συνάρτησης, επιστρέφονται οι δύο πίνακες που θα χρησιμοποιήσουμε για την επίλυση του συγκεκριμένου ζητήματος (θέμα 3), όταν αυτό ζητηθεί από το χρήστη στη κεντρική συνάρτηση της υλοποίησης.

Θέμα 1: Αλγόριθμος Ελάχιστου Μέσου Τετραγωνικού Σφάλματος - Least Mean Squares

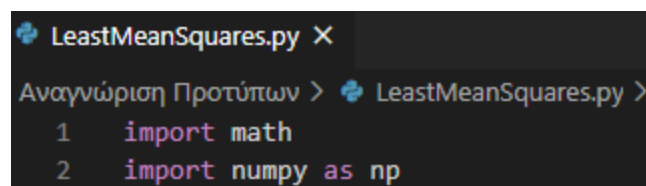
Στο σημείο αυτό και μετά την ανάλυση των βοηθητικών αρχείων, θα γίνει αναφορά στο πρώτο ζητούμενο της άσκησης που είναι η υλοποίηση του Αλγόριθμου Ελάχιστου Μέσου Τετραγωνικού Σφάλματος (Least Mean Squares), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την δοθέντα συνάρτηση διάκρισης για κάθε στοιχηματική εταιρεία και να αναγνωρίζει την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.

Το αρχείο που βρίσκεται η συνάρτηση για το πρώτο ερώτημα της άσκησης είναι το *LeastMeanSquares.py* και για την ανάπτυξη του αλγορίθμου κρίνεται αναγκαία η δημιουργία μίας Discriminant συνάρτησης με το λιγότερο δυνατό κόστος.

Ως τύπο της συνάρτησης έχουμε τον $y=w*x.T$ και ως βάρη για την καλύτερη ταξινόμηση των δεδομένων έχουμε τον τύπο εύρεσης των βαρών $w(n+1)=w(n)+\rho*e*x$. (2), με ρ = learning rate $e = y - y_i$ = error.

Στη παρούσα υλοποίηση χρησιμοποιήθηκε η μέθοδος – λογική One Against All, καθώς ο αλγόριθμος Least Mean Squares είναι ένας δυναδικός ταξινομητής (Binary Classifier) και για το λόγο αυτό δεν είναι δυνατό να ταξινομήσει πάνω από δύο τάξεις που έχουμε στο συγκεκριμένο παράδειγμα.

Ως εκ τούτου, ως λύση στο παραπάνω ζήτημα φτιάξαμε τρεις διαφορετικές καταστάσεις για κάθε αποτέλεσμα, το οποίο παρουσιάζεται ως ένας τρισδιάστατος πίνακας. Οι τρεις κατηγορίες που προαναφέρθηκαν είναι τα τρία πιθανά αποτελέσματα ενός ποδοσφαιρικού αγώνα, δηλαδή νίκη της γηπεδούχου ομάδας, ισοπαλία ή νίκη της φιλοξενούμενης ομάδας.



```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py >
1 import math
2 import numpy as np
```

Εικόνα Εφαρμογής 7

Ωστόσο, επειδή δεν μπορούμε να τις ταξινομήσουμε ταυτόχρονα και τις τρεις κατηγορίες θέτουμε ως “1” για μία από τις τρεις περιπτώσεις και μείον ένα τις υπόλοιπες.

Επομένως ο ταξινομητής μας θα ταξινομήσει τους 1 και τα -1 και στη προκειμένη περίπτωση τα -1 θα παρουσιάζουν δύο κατηγορίες μαζί.

Παρόλα αυτά, ακόμη δεν γνωρίζουμε ποια από τις τρεις κατηγορίες είναι πιο ωφέλιμη να πάρουμε, για το λόγο αυτό ο αλγόριθμος μας τρέχει για τις διαφορετικές καταστάσεις ως εξής:

- Νίκη της γηπεδούχου ομάδας +1, ενώ Ισοπαλία και Νίκη φιλοξενούμενου -1.
- Ισοπαλία +1, ενώ νίκη της γηπεδούχου και φιλοξενούμενης ομάδας -1.
- Νίκη της φιλοξενούμενης ομάδας, ενώ Ισοπαλία και Νίκη γηπεδούχου -1.

```
4  # Apply the hypothesis function (Wt*x).
5  def hypothesis(x, theta):
6
7      h = np.transpose(np.array(theta)).dot(np.array(x))
8      return h
9
10 # Find the cost function.
11 def costFunction(theta, x, y):
12
13     # Initialize variables.
14     factor = 1 / len(x)
15     sum = 0
16
17     # A For-Loop Statement, in order
18     # to calculate the sum.
19     for i in range(0, len(x)):
20         sum += math.pow((hypothesis(x[i], theta) - y[i]), 2)
21     return factor * sum
22
23 # Implement the w(n)+2*p*Error*x
24 def learnThetaSingle(theta, x, y, alpha):
25     return theta + 2*alpha * (y - hypothesis(x, theta)) * x
26
```

Εικόνα Εφαρμογής 8

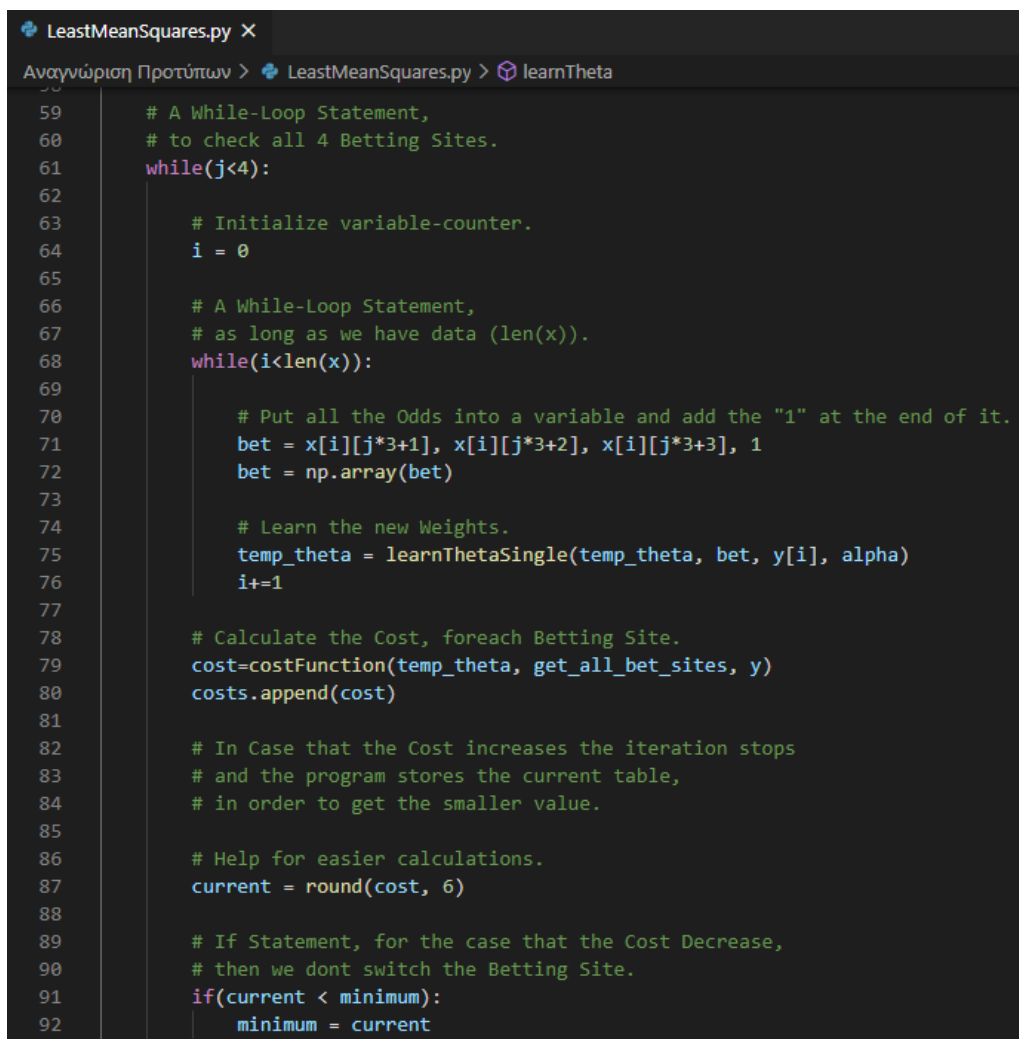
Επιπροσθέτως, είναι σημαντικό να τονίσουμε τη διαδικασία με την οποία θα βρεθούν τα βάρη της Συνάρτησης Διάκρισης (Discriminant Function). Ειδικότερα, κάθε φορά που θα τρέχουμε τον τύπο της συγκεκριμένης συνάρτησης θα βρίσκουμε διαφορετικά νέα βάρη, τα οποία αφού συγκεντρώσουμε όλα τα δεδομένα θα λαμβάνουμε ως κριτήριο διακοπής της επανάληψης το τελικό κόστος. Η κάθε επανάληψη θα σταματά στη περίπτωση που το τελικό κόστος αρχίσει να ανεβαίνει.

```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py > learnTheta
27 # Learn the rate function.
28 def learnTheta(theta, x, y):
29
30     # Initialize variables.
31     alpha = 1/len(x)
32     temp_theta = theta
33     bet = []
34     current = []
35     matrix_weights = []
36     costs = []
37     performance = []
38     minimum = 1
39
40     # Getting all Betting Sites.
41     get_all_bet_sites = [0 for i in range(0,len(x))]
42
43     # A For-Loop Statement, in which we create a table with all
44     # the best and the "1" at the end of it.
45     # The "1" its to help for the cost calculation process.
46     for i in range(0,len(x)):
47
48         current = x[i][1], x[i][2], x[i][3],1
49         current = np.array(current)
50         get_all_bet_sites[i] = current
51
52     # Set the iterations to 10 (loops),
53     # in order to see if the cost decrease.
54     iterations = 10
55
56     # Initialize Betting Site variable, j.
57     j = 0
58
```

Εικόνα Εφαρμογής 9

Την προαναφερθείσα διαδικασία την χρειαζόμαστε καθώς το τελικό κόστος κάποια στιγμή θα φτάσει σε ένα ελάχιστο. Σε αυτό το ελάχιστο, δηλαδή στη πιο χαμηλή τιμή που θα μπορούσε να λάβει καθώς η υλοποίησή μας δεν θα μπορούσε να ταξινομήσει καλύτερα τα δεδομένα και εκεί είναι το σημείο καμπής όπου θα αρχίσει να ανεβαίνει το τελικό κόστος. Στο σημείο αυτό σταματάει η επανάληψη και το πρόγραμμα επιστρέφει τα βάρη τα οποία βρήκε ενώ βρίσκονταν στη χαμηλότερη τιμή.

Σε πιο πρακτικό επίπεδο, η υλοποίησή μας ξεκινάει με την συνάρτηση **init**. Στη συγκεκριμένη συνάρτηση και στην παρακάτω εικόνα διακρίνουμε τις τρεις καταστάσεις που αναφέραμε παραπάνω (τα τρία πιθανά ενδεχόμενα ενός ποδοσφαιρικού αγώνα) και δημιουργούμε με αυτό το τρόπο τα νέα labels και στη συνέχεια καλούμε την **learnTheta**, η οποία θα τρέξει τον τύπο της συνάρτησης μέσω της **learnThetaSingle**, αφού πρώτα προσθέσει τον άσσο στο τέλος κάθε στοιχηματικής απόδοσης (η μεταβλητή X) προκειμένου να βρεθεί το βάρος w_0 , μέχρι να γίνει το τελικό κόστος ελάχιστο.



```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py > learnTheta
59     # A While-Loop Statement,
60     # to check all 4 Betting Sites.
61     while(j<4):
62
63         # Initialize variable-counter.
64         i = 0
65
66         # A While-Loop Statement,
67         # as long as we have data (len(x)).
68         while(i<len(x)):
69
70             # Put all the Odds into a variable and add the "1" at the end of it.
71             bet = x[i][j*3+1], x[i][j*3+2], x[i][j*3+3], 1
72             bet = np.array(bet)
73
74             # Learn the new Weights.
75             temp_theta = learnThetaSingle(temp_theta, bet, y[i], alpha)
76             i+=1
77
78         # Calculate the Cost, foreach Betting Site.
79         cost=costFunction(temp_theta, get_all_bet_sites, y)
80         costs.append(cost)
81
82         # In Case that the Cost increases the iteration stops
83         # and the program stores the current table,
84         # in order to get the smaller value.
85
86         # Help for easier calculations.
87         current = round(cost, 6)
88
89         # If Statement, for the case that the Cost Decrease,
90         # then we dont switch the Betting Site.
91         if(current < minimum):
92             minimum = current
```

Εικόνα Εφαρμογής 10

Το τελικό κόστος υπολογίζεται μέσω της συνάρτησης **costFunction**, η οποία υπολογίζει τον τύπο $(\sum w \cdot x.T - y)^2 / \text{len}(x)$.

Η συνάρτηση **hypothesis** χρησιμοποιεί ως μεταβλητές τις **αποδόσεις των στοιχηματικών** εταιριών και τα **labels** από την **learnTheta**. Η συγκεκριμένη συνάρτηση χρησιμοποιείται για να υπολογίζει τον πρώτο τύπο που αναφέρθηκε παραπάνω.

Ενώ, η συνάρτηση **learnTheta** επιστρέφει τον πίνακα με τα βάρη και το τελικό κόστος για κάθε στοιχηματική εταιρεία.

Στη συνέχεια η **init** θα εξετάσει με τα συγκεντρωτικά δεδομένα και θα βρει ποια στοιχηματική εταιρία έχει τα καλύτερα αποτελέσματα, την οποία και θα εμφανίσει στο χρήστη μέσω μηνύματος κονσόλας.

```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py > learnTheta
89     # If Statement, for the case that the Cost Decrease,
90     # then we dont switch the Betting Site.
91     if(current < minimum):
92         minimum = current
93
94     # Else, in case that the Cost Increases the program will
95     # save and store the current data (Weights, Costs etc.).
96     else:
97
98         # Change iteration value, in order to see
99         # if the cost will decrease in the
100        # next (only the next one) iteration.
101        iterations-=1
102
103        # If iteration are over.
104        if(iterations == 0):
105
106            matrix_weights.append(temp_theta)
107            temp_theta = [1,1,1,1]
108            j+=1
109
110            # Try/Catch Statement.
111            try:
112                for i in range(0,len(x)):
113                    get_all_bet_sites[i] = x[i][j*3+1], x[i][j*3+2], x[i][j*3+3],1
114                    performance.append(costs[np.argmin(costs)])
115            except:
116                performance.append(costs[np.argmin(costs)])
117
118            # Setting variables.
119            iterations = 10
120            minimum = 1
121            performance.append(costs[np.argmin(costs)])
122            costs = []
123
124        # Return the Weights' Table from all
125        # the Betting Sites, with their costs.
126        return matrix_weights,performance
```

Εικόνα Εφαρμογής 11

Η συνάρτηση **test** θα δείξει στο χρήστη πόσα σωστά και πόσα λάθος πέτυχε η υλοποίηση μετά τη διαδικασία του training αφού πρώτα έχει καταλήξει στη πιο εύστοχη στοιχηματική εταιρία.

```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py > learnTheta
128 # Init Function.
129 def init(theta,x,y):
130
131     # Create 3 Discrimibabt Functions,
132     # then we will ecide which one to keep.
133     # The y (labels) foreach function.
134     output = [[1,-1,-1],[-1,1,-1],[-1,-1,1]]
135
136     # Initialize variables.
137     new_matrix_weights = []
138     new_performance = []
139     new_y = [0 for i in range(0,len(y))]
140
141     # A For-Loop Statement in order to change the tables,
142     # when the function changes.
143     for j in output:
144         for i in range(0,len(y)):
145
146             if(y[i][1] == '1'):
147                 new_y[i] = j[0]
148
149             elif(y[i][1] == '0'):
150                 new_y[i] = j[1]
151
152             else:
153                 new_y[i] = j[2]
154
155     # The Least Mean Square Algorith will execute and check
156     # all the Betting Sites at the same time.
157     matrix_weights,performance=learnTheta(theta, x, new_y)
158
159     # The new Weights after the Algorithm execution foreach Betting Site.
160     new_matrix_weights.append(matrix_weights)
161
162     # The Cost after the Algorithm execution foreach Betting Site.
163     new_performance.append(performance)
```

Εικόνα Εφαρμογής 12

Ακολουθεί ένα ενδεικτικό παράδειγμα τρεξίματος αλγορίθμου της υλοποίησης, στο οποίο παρατηρούμε ότι εμφανίζεται το όνομα της στοιχηματικής εταιρείας η οποία είναι η πιο εύστοχη, η αναλογία από τα σωστά και τα λάθη τα βάρη και το τελικό κόστος, ως μήνυμα στη κονσόλα του εκάστοτε χρήστη.

```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py > learnTheta
165 # Initialize Variables.
166 minimum_performance = []
167 final_weights = []
168 final_performance = []
169 right_output = []
170
171 # A For-Loop for the 4 Betting Sites.
172 for i in range(0,4):
173
174     # Calculating the performance, foreach Betting Site.
175     minimum_performance = new_performance[0][i], new_performance[1][i], new_performance[2][i]
176     index = np.argmin(minimum_performance)
177     right_output.append(index)
178
179     # The Final Weight (based on smaller Cost, foreach Betting Site).
180     final_weights.append(new_matrix_weights[index][i])
181     # The Final Costs (the smallest ones).
182     final_performance.append(new_performance[index][i])
183
184 # The output that the program will use (keep) is the one with
185 # the more correctly predicted matches.
186 final_output = output[right_output[np.argmin(final_performance)]]
187
188 return final_weights,final_performance,final_output
```

Εικόνα Εφαρμογής 13

```
LeastMeanSquares.py X
Αναγνώριση Προτύπων > LeastMeanSquares.py > init
190 # Testing Function.
191 def test(theta,x,y,output_matrix):
192
193     # Create a Table with the best output, that the program found.
194     new_y = [0 for i in range(0,len(y))]
195
196     # A For-Loop Statement, as long as
197     # the Output's Table has data available.
198     for i in range(0,len(y)):
199
200         if(y[i][1] == '1'):
201             new_y[i] = output_matrix[0]
202
203         elif(y[i][1] == '0'):
204             new_y[i] = output_matrix[1]
205
206         else:
207             new_y[i] = output_matrix[2]
208
209     # Initialize Wrongs Counter.
210     wrongs_counter = 0
211     # Initialize Rights Counter.
212     rights_counter = 0
213
214     # An Inner For-Loop for the 4 Betting Sites.
215     for j in range(0,4):
216         for i in range(0,len(x)):
217
218             # Put all the Bets into a variable and add the "1" at the end of it.
219             bet = x[i][j*3+1], x[i][j*3+2], x[i][j*3+3], 1
220
221             # If the result is acceptable by the hypothetical function that we want, the count it as right.
222             if(np.sign(hypothesis(theta,bet)) == new_y[i]):
223                 rights_counter+=1
224
225             # Else If the result is not acceptable by the hypothetical function that we want, count it as wrong.
226             else:
227                 wrongs_counter+=1
228
229     # Returns the Right and Wrong - Predicted Matches.
230     return wrongs_counter,rights_counter
```

Εικόνα Εφαρμογής 14

Δυστυχώς η διαδικασία εύρεσης του ελάχιστου κόστους είναι αρκετά χρονοβόρα, ωστόσο επιστρέφει πάρα πολύ εύστοχα και αξιόπιστα αποτελέσματα.

Όπως φαίνεται στη παρακάτω εικόνα, η υλοποίηση βρίσκει σωστά περίπου το **75%** των αγώνων, ενώ πιο εύστοχη στοιχηματική είναι συνήθως η **IW** ή η **BET365**.

Δείτε τις παρακάτω ενδεικτικές εικόνες :

```
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Dimitris\Desktop\Αναγνώριση Προτύπων>C:/Python/python.exe "c:/Users/Dimitris/Desktop/Αναγνώριση Προτύπων/main.py"

Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.

Select either a,b,c or d option: a
```

Εικόνα Εφαρμογής 15

```
Process: Iteration No: 6
Please wait. The Training Process has been started!
Wrong - Predicted Matches: 539
Rights - Predicted Matches: 1441
Predicted Matches Accuracy: 72.7777777777777 %
Most Accurate Betting Site: IW
Weights: [ 0.15051436 0.12547031 -0.08625262 -0.92941971]
Completed: Iteration No: 6

Process: Iteration No: 7
Please wait. The Training Process has been started!
Wrong - Predicted Matches: 503
Rights - Predicted Matches: 1477
Predicted Matches Accuracy: 74.5959595959596 %
Most Accurate Betting Site: IW
Weights: [ 0.14957081 0.11921098 -0.08520181 -0.9118634 ]
Completed: Iteration No: 7

Process: Iteration No: 8
Please wait. The Training Process has been started!
Wrong - Predicted Matches: 585
Rights - Predicted Matches: 1395
Predicted Matches Accuracy: 70.4545454545454 %
Most Accurate Betting Site: B365
Weights: [ 0.17013535 -0.03894819 -0.0373987 -0.65125094]
Completed: Iteration No: 8
```

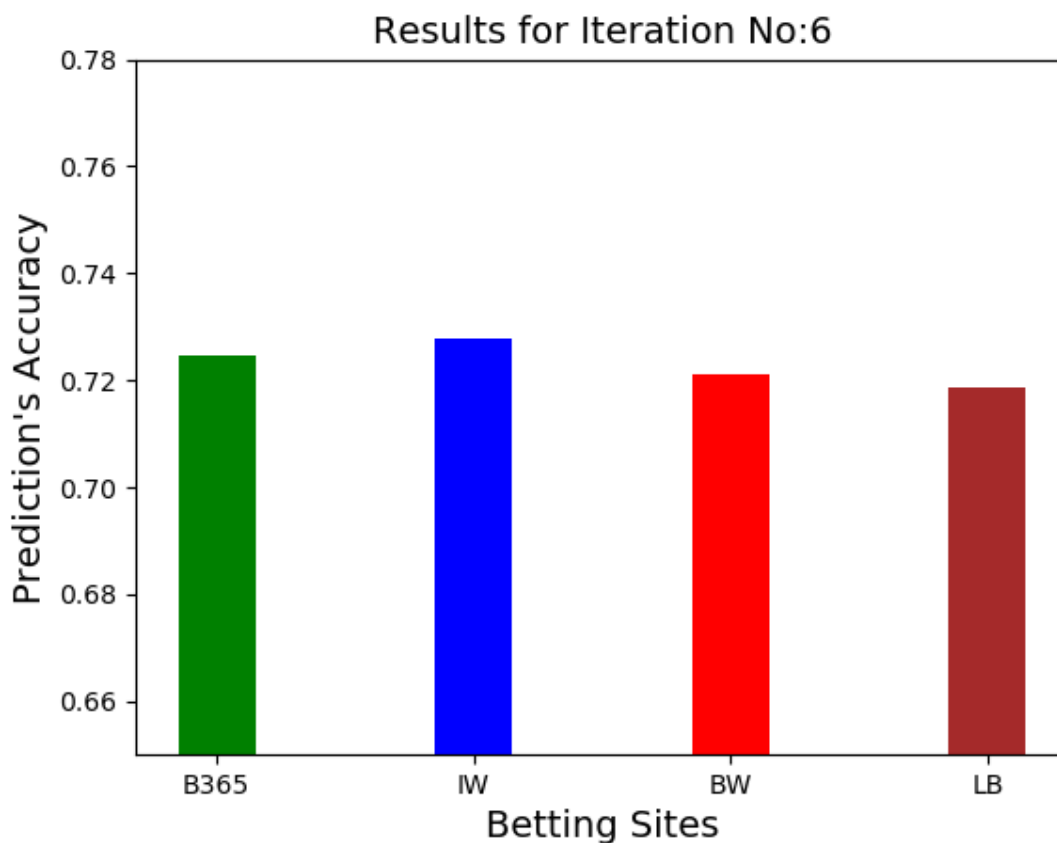
Εικόνα Εφαρμογής 16

Απεικόνιση Αποτελεσμάτων – Γραφήματα

Σε αυτή την ενότητα θα παρουσιάσουμε ενδεικτικά τα γραφήματα – plots που εμφανίζουμε για το εν λόγω ερώτημα.

Τα γραφήματα υλοποιούνται μέσω της βιβλιοθήκης **matplotlib.pyplot** και μέσα από το αρχείο *KFold_CrossValidation.py* και θα περιγραφούν σε επόμενη ενότητα.

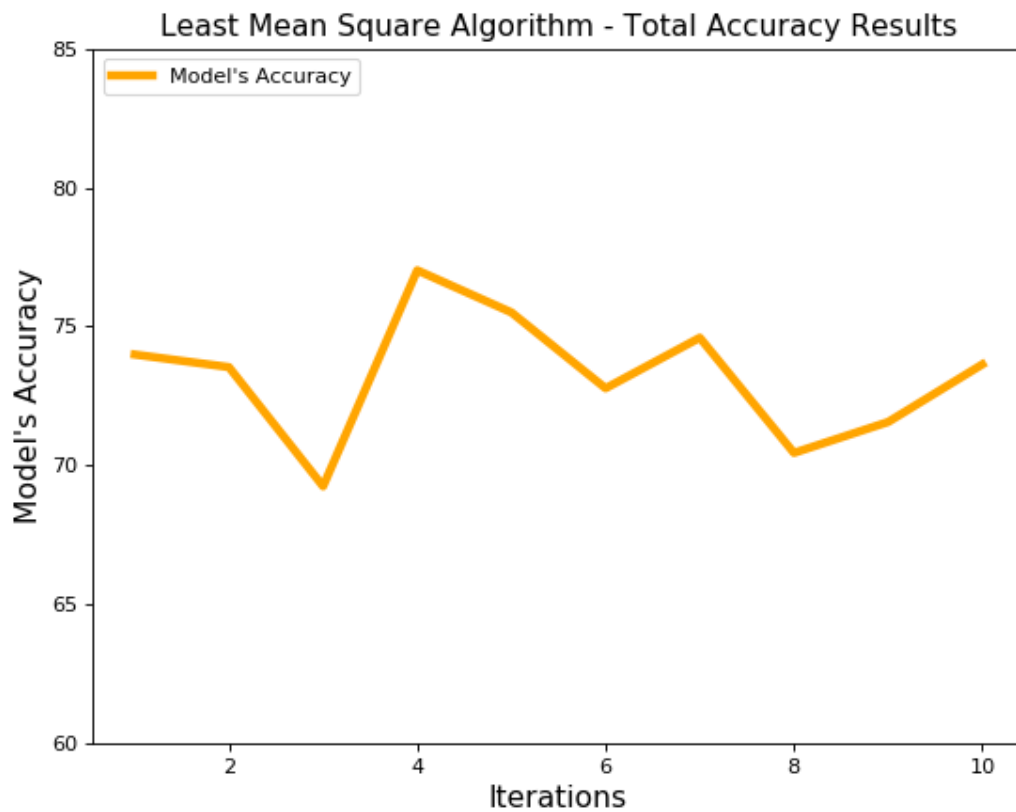
Μετά από κάθε επανάληψη (iteration) του K - Fold Cross Validation το πρόγραμμα θα σταματά και θα εμφανίζει ένα γράφημα με τα ποσοστά ευστοχίας των τεσσάρων στοιχηματικών, ακολουθεί ενδεικτική εικόνα :



Εικόνα Γραφήματος 1

Επιπλέον, μετά το πέρας των δέκα επαναλήψεων θα εμφανίζεται ένα ιστόγραμμα που θα δείχνει τα στατιστικά ευστοχίας της καλύτερης στοιχηματικής ανά επανάληψη.

Ακολουθούν σχετικές εικόνες :



Εικόνα Γραφήματος 2

Θέμα 2: Αλγόριθμος Ελάχιστου Τετραγωνικού Σφάλματος - Least Squares

Σε αυτή την ενότητα θα πραγματευτούμε την υλοποίηση του αλγορίθμου του Ελάχιστου Τετραγωνικού Σφάλματος – Least Squares.

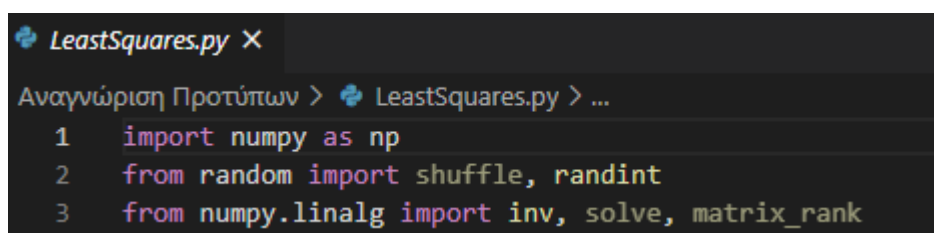
Ειδικότερα στο εν λόγω αρχείο κώδικα (αρχείο LeastSquares.py), υλοποιείται η λειτουργία του αλγορίθμου στη συγκεκριμένη περίπτωση αυτό γίνεται εφικτό με τη δημιουργία 5 μεθόδων – συναρτήσεων: train, test, predict, fixLabels και main.

Θα αναφερθούμε παρακάτω στις πέντε αυτές συναρτήσεις, καθώς στη παρούσα χρονική στιγμή θα αναφερθούμε στο κομμάτι της λογικής πίσω από την υλοποίηση και τις απαραίτητες βιβλιοθήκες για αυτή.

Πιο συγκεκριμένα, ο ταξινομητής Least Square αρχικά χωρίζει τα δεδομένα, που παίρνει από τη βάση σε δύο υποσύνολα, το σύνολο εκπαίδευσης (training set) και το σύνολο δοκιμής (testing set).

Το παραπάνω είναι απαραίτητο, ώστε ο αλγόριθμος που υλοποιείται να έχει τη δυνατότητα να κατανοεί καλύτερα πώς τα δεδομένα που εισάγουμε σχετίζονται με τις κλάσεις (επιβλεπόμενη μάθηση - supervised learning).

Πρωταρχικό βήμα είναι εισαγάγουμε τις απαραίτητες βιβλιοθήκες. Για τις μαθηματικές πράξεις μεταξύ πινάκων χρησιμοποιούμε τη βιβλιοθήκη **numpy**, για το ανακάτεμα των σειρών ενός πίνακα χρησιμοποιούμε τη **shuffle** και τη **randint**, από τη βιβλιοθήκη **random**. Ακολουθεί σχετική εικόνα από τον κώδικα :



```
LeastSquares.py X
Αναγνώριση Προτύπων > LeastSquares.py > ...
1 import numpy as np
2 from random import shuffle, randint
3 from numpy.linalg import inv, solve, matrix_rank
```

Εικόνα Εφαρμογής 17

Στη παρούσα χρονική στιγμή θα αναφερθούμε στις συναρτήσεις του δεύτερου ερωτήματος.

Καταρχάς στη συνάρτηση **train**, το **D** είναι το πλήθος των χαρακτηριστικών, ενώ το **K** είναι το πλήθος των κλάσεων. Έπειτα, αρχικοποιούμε τους πίνακες **sum1**, **sum2** και διαμορφώνουμε το διάνυσμα **x_i**, ώστε να είναι έτοιμο για τις πράξεις που θα ακολουθήσουν. Επιλύουμε ως προς το διάνυσμα βαρών, η εξίσωση του οποίου είναι υπεύθυνη για την εύρεση του και βρίσκεται στην απλοποιημένη μορφή της (Σχέση 3.43 του βιβλίου). Τέλος, στη συνάρτηση **test** (που θα δούμε παρακάτω) επιλύουμε την εξίσωση του Least Squares (Σχέση 3.45 στο βιβλίο), καλώντας την συνάρτηση **train**.

```
LeastSquares.py X
Αναγνώριση Προτύπων > LeastSquares.py > ...

5  # Train Function.
6  def train(x,y):
7
8      # Initialize shape's variables.
9      D = x.shape[1] + 1
10     K = y.shape[1]
11
12     # Initialize two matrices.
13     sum1 = np.zeros((D,D))
14     sum2 = np.zeros((D,K))
15
16     # Initialize variable.
17     i = 0
18
19     # xi*xi and xi*yi, is based to the
20     # Probability Density Function for "w".
21     # A For - Loop Statement
22     for x_i in x:
23
24         # Add array with "1", to the vector.
25         x_i = np.append(1, x_i)
26         # Getting y array.
27         y_i = y[i]
28
29         # xi * xi_hat - simplyfied version of
30         # the Function (P.P J(w) -> sxesi 3.43 apo to biblio).
31         sum1 += np.outer(x_i, x_i)
32         # xi * yi_hat - simplyfied version of
33         # the Function (P.P J(w) -> sxesi 3.43 apo to biblio).
34         sum2 += np.outer(x_i, y_i)
35
36         # Increase the counter, i.
37         i += 1
38
39     # Weights' vector: Multiply the sum2 with the inversed sum1.
40     # (sxesi 3.45 apo to biblio).
41     return np.dot(inv(sum1),sum2)
```

Εικόνα Εφαρμογής 18

Ακόμη καλούμε την **train** για να διαχωρίσουμε τα δεδομένα μας στα 2 υποσύνολα, εκπαίδευσης και δοκιμής, κατασκευάζουμε μια δομή επανάληψης *for* και καλούμε την **predict** για κάθε κλάση.

Η συνάρτηση **predict** προβλέπει την κλάση y για κάθε στοιχείο από τα δεδομένα μας κι επιστρέφει το αντίστοιχο αποτέλεσμα (πίνακας με τιμές 0 ή 1).

```
LeastSquares.py X
Αναγνώριση Προτύπων > LeastSquares.py > ...
43 # Predict Function.
44 def predict(W, x):
45
46     # Add the array of "1" ones,
47     # in order to do the operation.
48     x = np.append(1, x)
49
50     # Solve the  $W_{hat} * x$  .
51     values = list(np.dot(W.T, x))
52
53     # Finds maximum value.
54     winners = [i for i, x in enumerate(values) if x == max(values)]
55
56     # Choose randomly if the result is "0" or "1" (the y value).
57     index = randint(0, len(winners)-1)
58
59     # Setting the new variable winner and the value of "i" and "x", which are selected for the prediction.
60     winner = winners[index]
61
62     # Initialize a Zero Matrix.
63     y = [0 for x in values]
64     # Select value.
65     y[winner] = 1
66
67     return y
```

Εικόνα Εφαρμογής 19

Η συνάρτηση **test** ξεκινάει με την εκτέλεση του αλγορίθμου, με το μετρητή hits καταγράφουμε τις σωστά προβλεπόμενες κλάσεις - ετικέτες και τελικά με τη μεταβλητή accuracy υπολογίζουμε το συνολικό ποσοστό ακρίβειας του ταξινομητή που υλοποιήθηκε.

```
LeastSquares.py X
Αναγνώριση Προτύπων > LeastSquares.py > main
101 # Testing Function.
102 def test(a,b,c,d):
103
104     # Create Weights' vector.
105     W = train(a,b)
106
107     # Create Training Fields (Synola Ekpaideusis).
108     x = c
109     y = d
110
111     # Initialize variables.
112     total = y.shape[0]
113     i = 0
114     hits = 0
115
116     # A For-Loop Statement that if "i" from the
117     # "prediction" is equal to the value from
118     # the "y", then the Accuracy is increased.
119     # (If we predict correctly. ).
120     for i in range(total):
121
122         prediction = predict(W, x[i])
123         actual = list(y[i].A1)
124
125         # If accuracy hits correctly.
126         if prediction == actual:
127             hits += 1
128
129     # Then divide with its length and find the total Accuracy.
130     accuracy = hits/float(total)*100
131
132     # Console's Message, regarding the Predicted Matches' Accuracy stat appears.
133     print ("Predicted Matches Accuracy: " + str(accuracy) + "%", "(" + str(hits) + "/" + str(total) + ")")
```

Εικόνα Εφαρμογής 20

Ενώ, η συνάρτηση **fixLabels** πραγματοποιεί ορισμένες μορφοποιήσεις στις κλάσεις των ετικετών – outputs' labels ώστε να εκτελεστεί ομαλά ο αλγόριθμος.

```
LeastSquares.py X
Αναγνώριση Προτύπων > LeastSquares.py > ...
69 # Operation Function fixLabels
70 # (regarding outputs - labels of y).
71 def fixLabels(y):
72
73     # Initialize an empty array, newY.
74     newY = []
75
76     # A For - Loop, for length of the Labels' (y's) Data.
77     # Each list has the length of
78     # the class with the bigger value.
79     for i in range(len(y)):
80
81         # If Home Team wins, put 1 to the
82         # Betting Site's Odd for the home win
83         # and 0 to draw and away win.
84         if(y[i][1] == '1'):
85             newY.append([1,0,0])
86
87         # Else If the match ends as draw, put 1 to the
88         # Betting Site's Odd for draw and
89         # 0 to home and away wins.
90         elif(y[i][1]=='0'):
91             newY.append([0,1,0])
92
93         # Else If Away Team wins, put 1 to the
94         # Betting Site's Odd for the away win
95         # and 0 to home win and draw.
96         else:
97             newY.append([0,0,1])
98
99     return np.matrix(newY)
```

Εικόνα Εφαρμογής 21

Τελικά, καλούμε τη πέμπτη και τελευταία μέθοδο τη **main**, η οποία αρχικά εμφανίζει ένα μήνυμα έναρξης εκπαίδευσης μέσω της κονσόλας προς το χρήστη, παρέχει στους data και classes, τα δεδομένα και τις κλάσεις αντίστοιχα. Στη συνέχεια τους μετατρέπουμε σε πίνακες x, y, μέσω της βιβλιοθήκης **numpy** και καλούμε τη **fixLabels** για τις μορφοποιήσεις στις

κλάσεις μας, ενώ στο τέλος καλούμε την **test** και ξεκινάει η εκτέλεση του αλγορίθμου Ελάχιστου Τετραγωνικού Σφάλματος.

```
LeastSquares.py X
Αναγνώριση Προτύπων > LeastSquares.py > fixLabels
135 # Main Least Square Function.
136 def main(x_train,y_train,x_test,y_test):
137
138     # Console's Message, regarding the training process, appears.
139     # print('Please wait. The Training Process has been started!')
140
141     # Initialize Variables.
142     data = x_train[:,1:]
143     classes = y_train
144
145     # Convert input data to matrices, via numpy.
146     x = np.matrix(data)
147     # Reform y for the next operations.
148     y = fixLabels(classes)
149     # Convert input data to matrices, via numpy.
150     c = np.matrix(x_test[:,1:])
151     # Reform y for the next operations.
152     d = fixLabels(y_test)
153
154     # Data Shuffle for better precision.
155     # Initialize, a temp table, in which
156     # we put both x abd y.
157     z = []
158
159     # Getting size.
160     # size-1, because the
161     # For-Loop starts ftom 0.
162     size = x.shape[0] - 1
163
164     # A For-Loop Statement, regarding size.
165     for i in range(size):
166
167         z.append((x[i],y[i]))
168
169     # A For-Loop Statement, regarding size.
170     for i in range(size):
171         x[i] = z[i][0]
172         y[i] = z[i][1]
173
174     # Append the Training and Testing Data.
175     test(x,y,c,d)
```

Εικόνα Εφαρμογής 22

Επιπροσθέτως, έχουμε τη δυνατότητα να κάνουμε shuffle του πίνακα (τυχαία αλλαγή σειρών) στα δεδομένα μας για καλύτερη ακρίβεια των αποτελεσμάτων της υλοποίησης.

Τέλος, με την ίδια διαδικασία με το προηγούμενο ερώτημα βρίσκουμε την στοιχηματική εταιρία με τα καλύτερα ποσοστά ευστοχίας και μέσω αυτού θα δημιουργείτε το σχετικό γράφημα.

Εκτελώντας το συγκεκριμένο ερώτημα από το μενού επιλογών λαμβάνουμε τα παρακάτω αποτελέσματα :

```
C:\Users\Dimitris\Desktop\Αναγνώριση Προτύπων>C:/Python/python.exe "c:/Users/Dimitris/Desktop/Αναγνώριση Προτύπων/main.py"

Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.

Select either a,b,c or d option: b
```

Εικόνα Εφαρμογής 23

```
Process: Iteration No: 7
Please wait. The Training Process has been started!
Wrong - Predicted Matches: 235
Rights - Predicted Matches: 260
Predicted Matches Accuracy: 52.52525252525253 %
Most Accurate Betting Site: IW
Completed: Iteration No: 7
```

Εικόνα Εφαρμογής 24

```
Process: Iteration No: 8
Please wait. The Training Process has been started!
Wrong - Predicted Matches: 205
Rights - Predicted Matches: 290
Predicted Matches Accuracy: 58.58585858585859 %
Most Accurate Betting Site: B365
Completed: Iteration No: 8
```

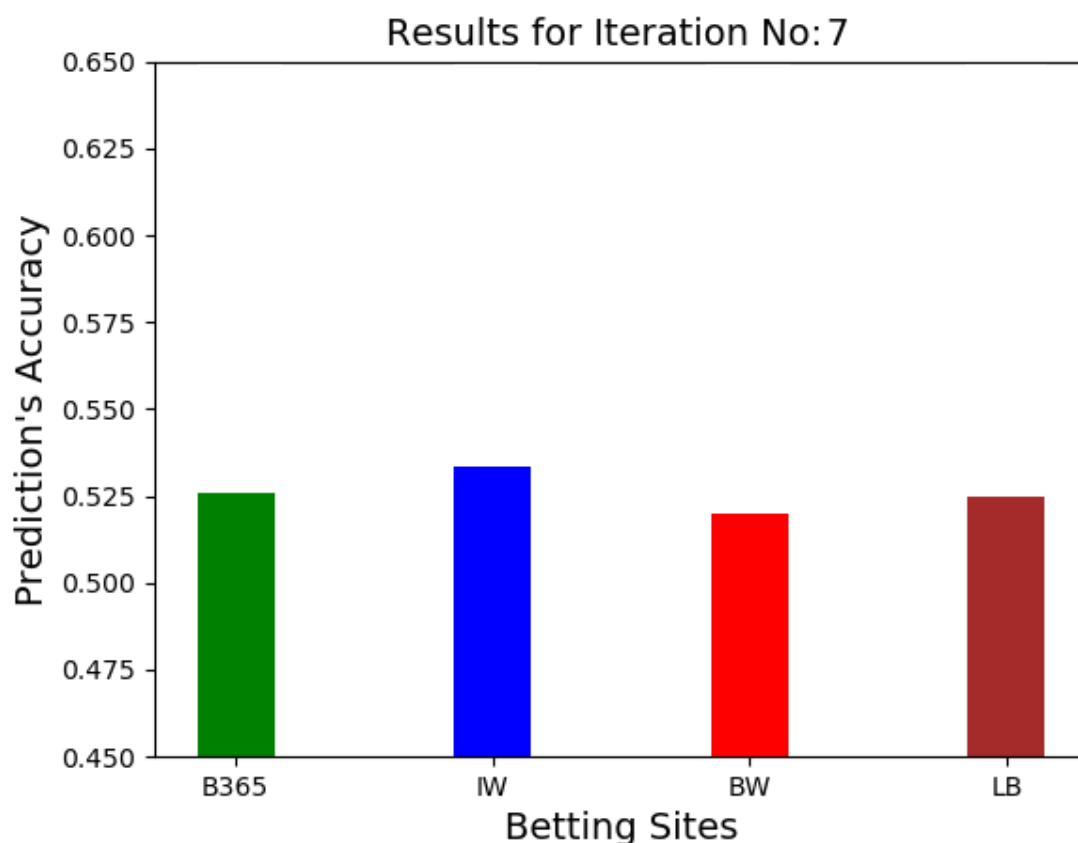
Εικόνα Εφαρμογής 25

Απεικόνιση Αποτελεσμάτων – Γραφήματα

Στη συγκεκριμένα ενότητα θα παρουσιάσουμε ενδεικτικά τα γραφήματα – plots που εμφανίζουμε για το δεύτερο ερώτημα της εργασίας.

Τα γραφήματα υλοποιούνται μέσω της βιβλιοθήκης **matplotlib.pyplot** και μέσα από το αρχείο *KFold_CrossValidation.py* και θα περιγραφούν σε επόμενη ενότητα.

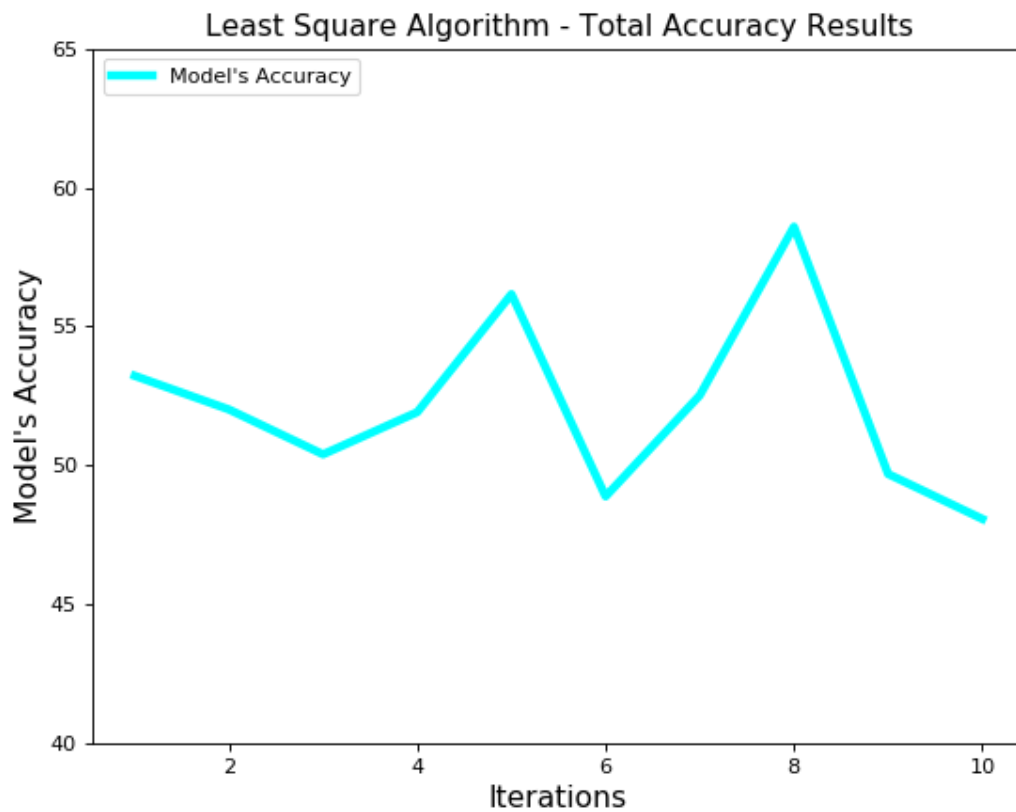
Μετά από κάθε επανάληψη (iteration) του K - Fold Cross Validation το πρόγραμμα θα σταματά και θα εμφανίζει ένα γράφημα με τα ποσοστά ευστοχίας των τεσσάρων στοιχηματικών, ακολουθεί ενδεικτική εικόνα :



Εικόνα Γραφήματος 3

Επιπλέον, μετά το πέρας των δέκα επαναλήψεων θα εμφανίζεται ένα ιστόγραμμα που θα δείχνει τα στατιστικά ευστοχίας της καλύτερης στοιχηματικής ανά επανάληψη.

Ακολουθούν σχετικές εικόνες :



Εικόνα Γραφήματος 4

Θέμα 3: Πολυστρωματικό Νευρωνικό Δίκτυο – MultiLayer Neutral Network

Στο τελευταίο ερώτημα της παρούσας υλοποίησης καλούμαστε να υλοποιήσετε ένα Πολυστρωματικό Νευρωνικό Δίκτυο (Multi-Layer Neutral Network).

Το αρχείο που υπάρχει η υλοποίηση του Νευρωνικού Δικτύου, είναι το MultiLayerNN.py. Το πολυστρωματικό νευρωνικό δίκτυο το οποίο δημιουργείται θα ταξινομεί σε τρεις κατηγορίες. Για αυτή την περίπτωση όπως έχουμε προ αναφέρει (στην ενότητα [Μορφοποίηση Παρεχόμενης Βάσης Δεδομένων](#)) στη συνάρτηση **init** καλείται πρώτα η **MNN_data** η οποία θα διαμορφώσει κατάλληλα τα δεδομένα προκειμένου οι τρεις κατηγορίες να διατυπώνονται ως εξής

$$[0,1,2] = [0,1,0], [1,0,0], [0,0,1]$$

ενώ ταυτόχρονα χρησιμεύει και για να κάνουμε concerate τα δεδομένα.

Για να γίνει αυτό πρέπει να εισάγουμε τις παρακάτω βιβλιοθήκες τις Python:

```
MultiLayerNN.py > init
1  import numpy as np
2  import tensorflow as tf
3  from keras.models import Sequential
4  from keras.layers.core import Dense
5  from keras.models import Sequential
6  from keras.layers import Dense,Dropout
7  from keras.regularizers import l2
8  from tensorflow import keras
9  from tensorflow.keras import layers
10 import matplotlib.pyplot as plt
```

Εικόνα Εφαρμογής 26

Έπειτα, καλείται η συνάρτηση **init**, η οποία δημιουργεί το μοντέλο του Νευρωνικού Δικτύου, το οποίο έχει χτίσει μέσω της βιβλιοθήκης **tensorflow** και **keras**.

Το μοντέλο της υλοποίησης μετά από τις απαραίτητες δοκιμές είναι το ακόλουθο, καθώς αυτό είχε τα καλύτερα αποτελέσματα στη διαδικασία της εκπαίδευσης - training.

```
12 # Function, which creates the neurons.
13 def init(x_train,y_train,x_test,y_test):
14
15     # Create neuron model via keras library.
16     model = keras.Sequential()
17     model.add(layers.Dense(4*28, input_dim=28, activation='relu'))
18     model.add(layers.Dropout(0.5))
19     model.add(layers.Dense(4*28,activation='relu'))
20     model.add(layers.Dense(2*28, activation='relu'))
21     model.add(layers.Dense(3, activation='softmax'))
22
23     # kernel_regularizer=l2(0.0001),
24     # activity_regularizer=l2(0.01)
25
26     # Transforms the variables to an acceptable format.
27     x_train = x_train.astype("float32")
28     x_test = x_test.astype("float32")
29     y_train = y_train.astype("float32")
30     y_test = y_test.astype("float32")
31
32     # Compile and fit the keras model.
33     model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics= ['accuracy'])
34     history = model.fit(x_train, y_train, batch_size=512, epochs=1000, verbose=1, validation_data = (x_test, y_test))
35
36     # Put all the model's predictions into a table.
37     predictions = model.predict(x_test).round()
38
39     # Initialize the rights and wrong - predicted
40     # matches'counter.
41     rights_counter = 0
42     wrongs_counter = 0
```

Εικόνα Εφαρμογής 27

Ένα πρόβλημα που προέκυψε στη παραπάνω υλοποίηση ήταν το *overfitting*. Ειδικότερα, το Πολυστρωματικό Νευρωνικό Δίκτυο εκπαιδεύεται τόσο καλά πάνω στο συγκεκριμένο σύνολο δεδομένων (training data), ώστε να μην μπορεί να ταξινομήσει τα test_data.

Κάνοντας ωστόσο προσθήκη ένα dropout, στο μοντέλο μας, το οποίο έχει σκοπό να αφαιρεί τυχαία κάποια nodes, αυξάνει το *validation accuracy*.

Για loss function χρησιμοποιήσαμε από τις δυνατές επιλογές της **keras**, το categorical_crossentropy, ως optimizer τον adam και καταλήξαμε στις χίλιες επαναλήψεις (iterations) γιατί με αυτό το συνδυασμό το μοντέλο έχει το καλύτερο δυνατό *training* με το λιγότερο δυνατό *overfitting*.

Στο τέλος μετράει τα σωστά και τα λάθη και τα εμφανίζει αφού πρώτα τα έχει κρατήσει σε πίνακα τις προβλέψεις που έκανε.


```
44     # A For-Loop, which gets the right
45     # and the wrong - predicted matches.
46     for i,j in enumerate(predictions):
47
48         # If the match is predicted correctly.
49         if((y_test[i] == j).all()):
50             rights_counter+=1
51
52         # Else If the match is predicted incorrectly.
53         else:
54             wrongs_counter+=1
55
```

Εικόνα Εφαρμογής 28

Εκτελώντας το μοντέλο στα αποτελέσματά του βλέπουμε ότι εμφανίζεται κάθε επανάληψη (iteration) και πώς η εκπαίδευση γίνεται καλύτερη αλλά από ένα σημείο και μετά βλέπουμε το validation accuracy να μειώνεται.

Ωστόσο δεν μπορούμε να σταματήσουμε το πρόγραμμα και την εκτέλεσή του εξαιτίας της χαμηλής εκπαίδευση πάλι θα έχουμε ένα παρόμοιο αποτέλεσμα.

Ως εκ τούτου, ορίζοντας τις επαναλήψεις του προγράμματος στις χίλιες επαναλήψεις υπάρχει εξισορρόπηση των αποτελεσμάτων.

```
56     # Making Multi-Layer Neural Network Plots.
57     # Summarize history for Accuracy.
58     plt.plot(history.history['accuracy'])
59     plt.plot(history.history['val_accuracy'])
60     plt.title('Multi-Layer Neural Network Accuracy')
61     plt.ylabel('Accuracy')
62     plt.xlabel('Epoch')
63     plt.legend(['Train Accuracy', 'Test Accuracy'], loc='upper left')
64     plt.show()
65
66     # Summarize history for loss.
67     plt.plot(history.history['loss'])
68     plt.plot(history.history['val_loss'])
69     plt.title('Multi-Layer Neural Network Loss')
70     plt.ylabel('Loss')
71     plt.xlabel('Epoch')
72     plt.legend(['Train Accuracy', 'Test Accuracy'], loc='upper right')
73     plt.show()
74
75     # Console's Message, regarding the Wrong - Predicted Matches appears.
76     print('Wrong - Predicted Matches: ' + str(wrongs_counter))
77     # Console's Message, regarding the Right - Predicted Matches appears.
78     print('Rights - Predicted Matches: ' + str(rights_counter))
79     # Console's Message, regarding the Predicted Matches' Accuracy stat appears.
80     print('Predicted Matches Accuracy: ' + str(rights_counter/(wrongs_counter + rights_counter)))
```

Εικόνα Εφαρμογής 29

Τέλος, μέσω της βιβλιοθήκης **matplotlib**, υλοποιούνται τα γραφήματα του συγκεκριμένου αλγορίθμου

Ακολουθούν ενδεικτικά στιγμιότυπα από την εκτέλεση του προγράμματος :

```
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Dimitris\Desktop\Αναγνώριση Προτύπων>C:\Python\python.exe "c:/Users/Dimitris/Desktop/Αναγνώριση Προτύπων/main.py"

Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.

Select either a,b,c or d option: c
```

Εικόνα Εφαρμογής 30

```
Process: Iteration No: 3
Please wait. The Training Process has been started!
Epoch 1/1000
6/6 [=====] - 0s 24ms/step - loss: 11.8232 - accuracy: 0.4155 - val_loss: 7.5308 - val_accuracy: 0.2585
Epoch 2/1000
6/6 [=====] - 0s 5ms/step - loss: 6.3726 - accuracy: 0.3496 - val_loss: 2.0915 - val_accuracy: 0.3385
Epoch 3/1000
6/6 [=====] - 0s 6ms/step - loss: 4.8867 - accuracy: 0.3745 - val_loss: 4.1865 - val_accuracy: 0.3046
Epoch 4/1000
6/6 [=====] - 0s 6ms/step - loss: 3.8984 - accuracy: 0.3244 - val_loss: 2.0177 - val_accuracy: 0.3138
Epoch 5/1000
6/6 [=====] - 0s 6ms/step - loss: 3.1475 - accuracy: 0.3864 - val_loss: 1.9225 - val_accuracy: 0.3108
Epoch 6/1000
6/6 [=====] - 0s 6ms/step - loss: 2.6768 - accuracy: 0.3507 - val_loss: 2.4040 - val_accuracy: 0.2492
Epoch 7/1000
6/6 [=====] - 0s 5ms/step - loss: 2.3016 - accuracy: 0.3708 - val_loss: 1.7429 - val_accuracy: 0.2677
Epoch 8/1000
6/6 [=====] - 0s 6ms/step - loss: 2.0133 - accuracy: 0.3962 - val_loss: 1.7254 - val_accuracy: 0.2677
Epoch 9/1000
6/6 [=====] - 0s 5ms/step - loss: 1.9644 - accuracy: 0.3641 - val_loss: 1.8038 - val_accuracy: 0.2585
Epoch 10/1000
6/6 [=====] - 0s 6ms/step - loss: 1.7538 - accuracy: 0.3736 - val_loss: 1.3561 - val_accuracy: 0.2800
Epoch 11/1000
6/6 [=====] - 0s 6ms/step - loss: 1.5986 - accuracy: 0.4050 - val_loss: 1.3607 - val_accuracy: 0.2585
Epoch 12/1000
6/6 [=====] - 0s 6ms/step - loss: 1.5138 - accuracy: 0.3769 - val_loss: 1.2543 - val_accuracy: 0.2800
Epoch 13/1000
6/6 [=====] - 0s 9ms/step - loss: 1.4351 - accuracy: 0.3863 - val_loss: 1.1699 - val_accuracy: 0.3108
Epoch 14/1000
6/6 [=====] - 0s 7ms/step - loss: 1.3682 - accuracy: 0.3960 - val_loss: 1.1524 - val_accuracy: 0.3169
Epoch 15/1000
6/6 [=====] - 0s 5ms/step - loss: 1.3400 - accuracy: 0.3857 - val_loss: 1.1539 - val_accuracy: 0.3108
Epoch 16/1000
6/6 [=====] - 0s 6ms/step - loss: 1.3037 - accuracy: 0.3947 - val_loss: 1.1030 - val_accuracy: 0.3877
Epoch 17/1000
6/6 [=====] - 0s 5ms/step - loss: 1.2656 - accuracy: 0.3951 - val_loss: 1.0863 - val_accuracy: 0.3969
Epoch 18/1000
6/6 [=====] - 0s 5ms/step - loss: 1.2409 - accuracy: 0.4053 - val_loss: 1.1078 - val_accuracy: 0.3508
Epoch 19/1000
6/6 [=====] - 0s 4ms/step - loss: 1.2119 - accuracy: 0.4141 - val_loss: 1.0941 - val_accuracy: 0.3723
Epoch 20/1000
```

Εικόνα Εφαρμογής 31

```
Epoch 73/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0079 - accuracy: 0.4955 - val_loss: 1.0136 - val_accuracy: 0.5185
Epoch 74/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0164 - accuracy: 0.4830 - val_loss: 1.0137 - val_accuracy: 0.5123
Epoch 75/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0062 - accuracy: 0.5013 - val_loss: 1.0125 - val_accuracy: 0.5123
Epoch 76/1000
6/6 [=====] - 0s 6ms/step - loss: 1.0116 - accuracy: 0.4828 - val_loss: 1.0130 - val_accuracy: 0.5031
Epoch 77/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0085 - accuracy: 0.4932 - val_loss: 1.0136 - val_accuracy: 0.5154
Epoch 78/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0150 - accuracy: 0.4786 - val_loss: 1.0141 - val_accuracy: 0.5093
Epoch 79/1000
6/6 [=====] - 0s 7ms/step - loss: 1.0208 - accuracy: 0.4796 - val_loss: 1.0127 - val_accuracy: 0.5247
Epoch 80/1000
6/6 [=====] - 0s 4ms/step - loss: 1.0109 - accuracy: 0.4715 - val_loss: 1.0109 - val_accuracy: 0.5185
Epoch 81/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0080 - accuracy: 0.4919 - val_loss: 1.0110 - val_accuracy: 0.5031
Epoch 82/1000
6/6 [=====] - 0s 6ms/step - loss: 1.0151 - accuracy: 0.4911 - val_loss: 1.0106 - val_accuracy: 0.5031
Epoch 83/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0045 - accuracy: 0.4974 - val_loss: 1.0123 - val_accuracy: 0.5154
Epoch 84/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0125 - accuracy: 0.4919 - val_loss: 1.0122 - val_accuracy: 0.5123
Epoch 85/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0069 - accuracy: 0.5084 - val_loss: 1.0117 - val_accuracy: 0.5031
Epoch 86/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0044 - accuracy: 0.4899 - val_loss: 1.0120 - val_accuracy: 0.4877
Epoch 87/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0030 - accuracy: 0.4937 - val_loss: 1.0145 - val_accuracy: 0.5062
Epoch 88/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9971 - accuracy: 0.5088 - val_loss: 1.0134 - val_accuracy: 0.5062
Epoch 89/1000
6/6 [=====] - 0s 6ms/step - loss: 1.0113 - accuracy: 0.4852 - val_loss: 1.0145 - val_accuracy: 0.5123
Epoch 90/1000
6/6 [=====] - 0s 5ms/step - loss: 1.0103 - accuracy: 0.4846 - val_loss: 1.0112 - val_accuracy: 0.5093
```

Εικόνα Εφαρμογής 32

```
Epoch 600/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9037 - accuracy: 0.5761 - val_loss: 1.1116 - val_accuracy: 0.4722
Epoch 601/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9004 - accuracy: 0.5750 - val_loss: 1.1168 - val_accuracy: 0.4537
Epoch 602/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8997 - accuracy: 0.5752 - val_loss: 1.1080 - val_accuracy: 0.4537
Epoch 603/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9064 - accuracy: 0.5633 - val_loss: 1.1064 - val_accuracy: 0.4691
Epoch 604/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8871 - accuracy: 0.5692 - val_loss: 1.1176 - val_accuracy: 0.4475
Epoch 605/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8962 - accuracy: 0.5781 - val_loss: 1.1330 - val_accuracy: 0.4815
Epoch 606/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9075 - accuracy: 0.5596 - val_loss: 1.1303 - val_accuracy: 0.4907
Epoch 607/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8782 - accuracy: 0.5803 - val_loss: 1.1284 - val_accuracy: 0.4568
Epoch 608/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9107 - accuracy: 0.5589 - val_loss: 1.1277 - val_accuracy: 0.4599
Epoch 609/1000
6/6 [=====] - 0s 4ms/step - loss: 0.9027 - accuracy: 0.5701 - val_loss: 1.1266 - val_accuracy: 0.4630
Epoch 610/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8957 - accuracy: 0.5770 - val_loss: 1.1328 - val_accuracy: 0.4846
Epoch 611/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9212 - accuracy: 0.5471 - val_loss: 1.1156 - val_accuracy: 0.4660
Epoch 612/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9056 - accuracy: 0.5623 - val_loss: 1.1204 - val_accuracy: 0.4599
Epoch 613/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9031 - accuracy: 0.5765 - val_loss: 1.1266 - val_accuracy: 0.4691
Epoch 614/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9059 - accuracy: 0.5564 - val_loss: 1.1211 - val_accuracy: 0.4784
Epoch 615/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8964 - accuracy: 0.5631 - val_loss: 1.1270 - val_accuracy: 0.4846
Epoch 616/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9104 - accuracy: 0.5588 - val_loss: 1.1137 - val_accuracy: 0.4660
Epoch 617/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8973 - accuracy: 0.5794 - val_loss: 1.1149 - val_accuracy: 0.4630
Epoch 618/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9049 - accuracy: 0.5712 - val_loss: 1.1062 - val_accuracy: 0.4475
Epoch 619/1000
6/6 [=====] - 0s 5ms/step - loss: 0.9005 - accuracy: 0.5688 - val_loss: 1.1199 - val_accuracy: 0.4537
Epoch 620/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8930 - accuracy: 0.5801 - val_loss: 1.1321 - val_accuracy: 0.4599
```

Εικόνα Εφαρμογής 33

```
Epoch 985/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8444 - accuracy: 0.6002 - val_loss: 1.0344 - val_accuracy: 0.4800
Epoch 986/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8553 - accuracy: 0.5884 - val_loss: 1.0289 - val_accuracy: 0.5015
Epoch 987/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8448 - accuracy: 0.6008 - val_loss: 1.0386 - val_accuracy: 0.4738
Epoch 988/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8360 - accuracy: 0.6087 - val_loss: 1.0313 - val_accuracy: 0.4800
Epoch 989/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8345 - accuracy: 0.6027 - val_loss: 1.0310 - val_accuracy: 0.4985
Epoch 990/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8266 - accuracy: 0.6252 - val_loss: 1.0212 - val_accuracy: 0.4892
Epoch 991/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8303 - accuracy: 0.6042 - val_loss: 1.0297 - val_accuracy: 0.4738
Epoch 992/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8375 - accuracy: 0.6067 - val_loss: 1.0380 - val_accuracy: 0.4769
Epoch 993/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8277 - accuracy: 0.6008 - val_loss: 1.0358 - val_accuracy: 0.4892
Epoch 994/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8229 - accuracy: 0.6145 - val_loss: 1.0325 - val_accuracy: 0.4831
Epoch 995/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8257 - accuracy: 0.6020 - val_loss: 1.0330 - val_accuracy: 0.4831
Epoch 996/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8369 - accuracy: 0.6036 - val_loss: 1.0324 - val_accuracy: 0.4862
Epoch 997/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8369 - accuracy: 0.5917 - val_loss: 1.0392 - val_accuracy: 0.4769
Epoch 998/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8287 - accuracy: 0.6188 - val_loss: 1.0280 - val_accuracy: 0.4923
Epoch 999/1000
6/6 [=====] - 0s 5ms/step - loss: 0.8525 - accuracy: 0.5974 - val_loss: 1.0262 - val_accuracy: 0.4954
Epoch 1000/1000
6/6 [=====] - 0s 4ms/step - loss: 0.8369 - accuracy: 0.6025 - val_loss: 1.0369 - val_accuracy: 0.4892
Wrong - Predicted Matches: 216
Rights - Predicted Matches: 109
Predicted Matches Accuracy: 0.3353846153846154
```

Εικόνα Εφαρμογής 34

Απεικόνιση Αποτελεσμάτων – Γραφήματα

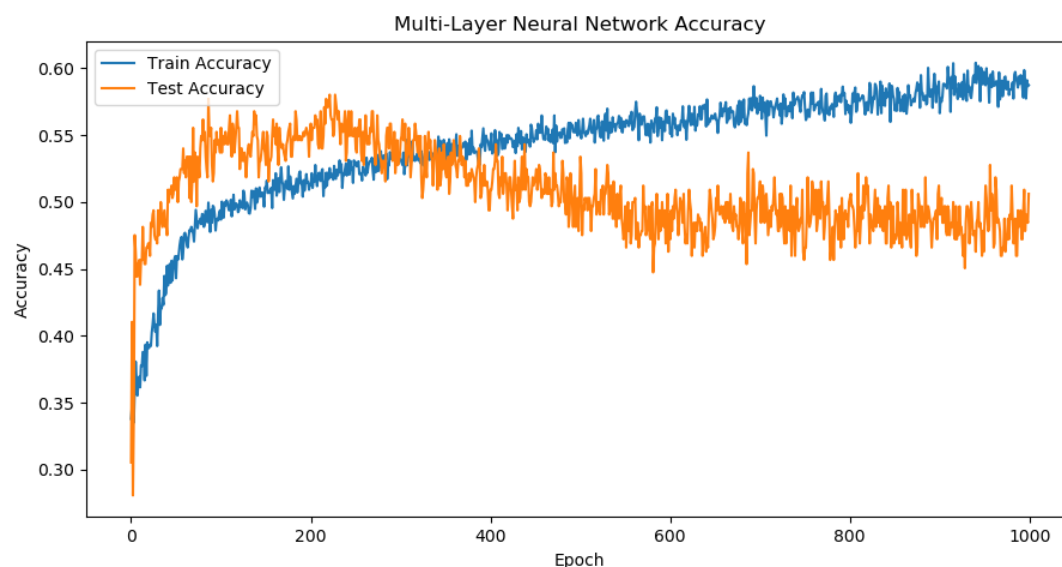
Στη παρούσα ενότητα θα παρουσιάσουμε ενδεικτικά τα γραφήματα – plots που εμφανίζουμε για το δεύτερο ερώτημα της εργασίας.

Τα γραφήματα υλοποιούνται μέσω της βιβλιοθήκης `matplotlib.pyplot` και μέσα από το αρχείο `KFold_CrossValidation.py` και θα περιγραφούν σε επόμενη ενότητα.

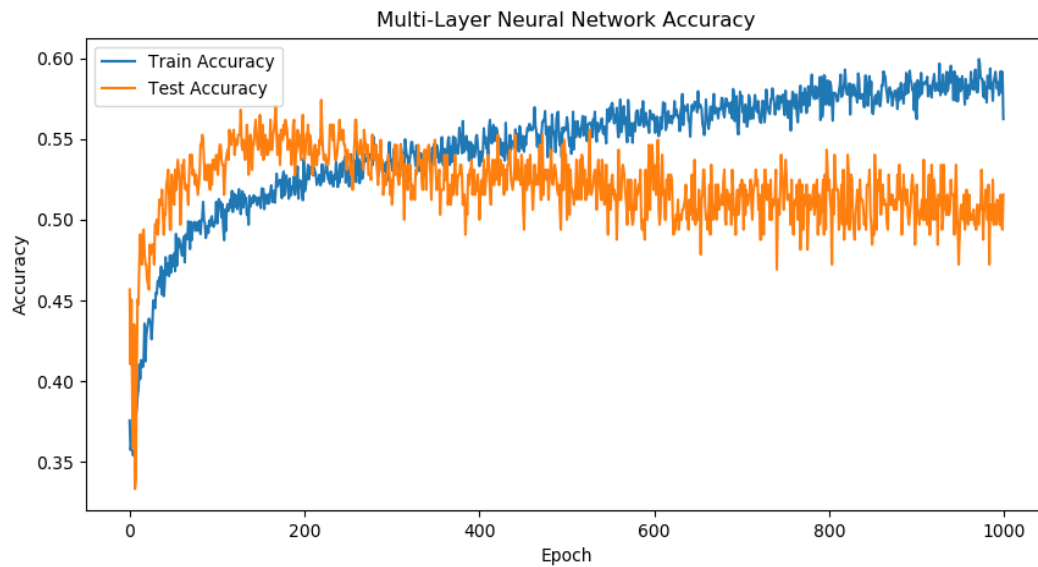
Μετά από κάθε επανάληψη (iteration) του K - Fold Cross Validation το πρόγραμμα θα σταματά και θα εμφανίζει ένα γράφημα που θα παρουσιάζει τη συνολική ευστοχία του μοντέλου και τη συνάρτηση του σφάλματος – λάθους ανά τις 1000 εποχές – επαναλήψεις μάθησης.

Ακολουθούν σχετικές εικόνες και από τις δύο κατηγορίες γραφημάτων που εμφανίζονται:

Εικόνα πορείας ευστοχίας – πρόβλεψης:

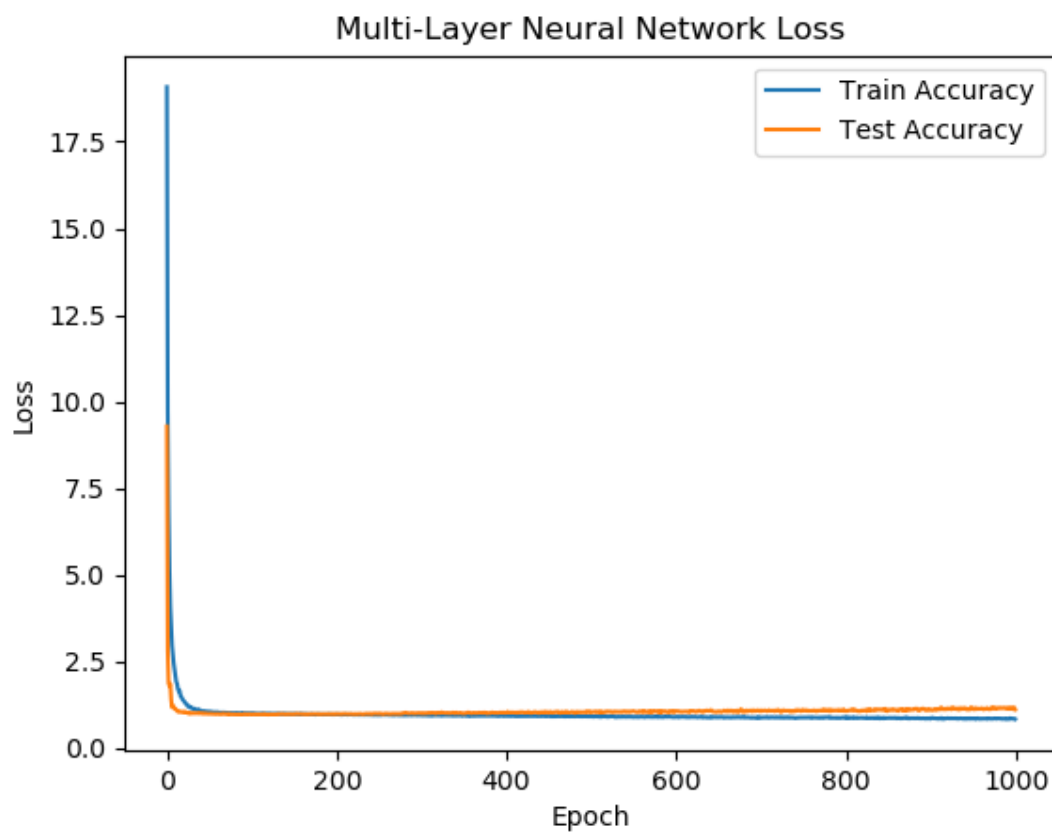


Εικόνα Γραφήματος 5

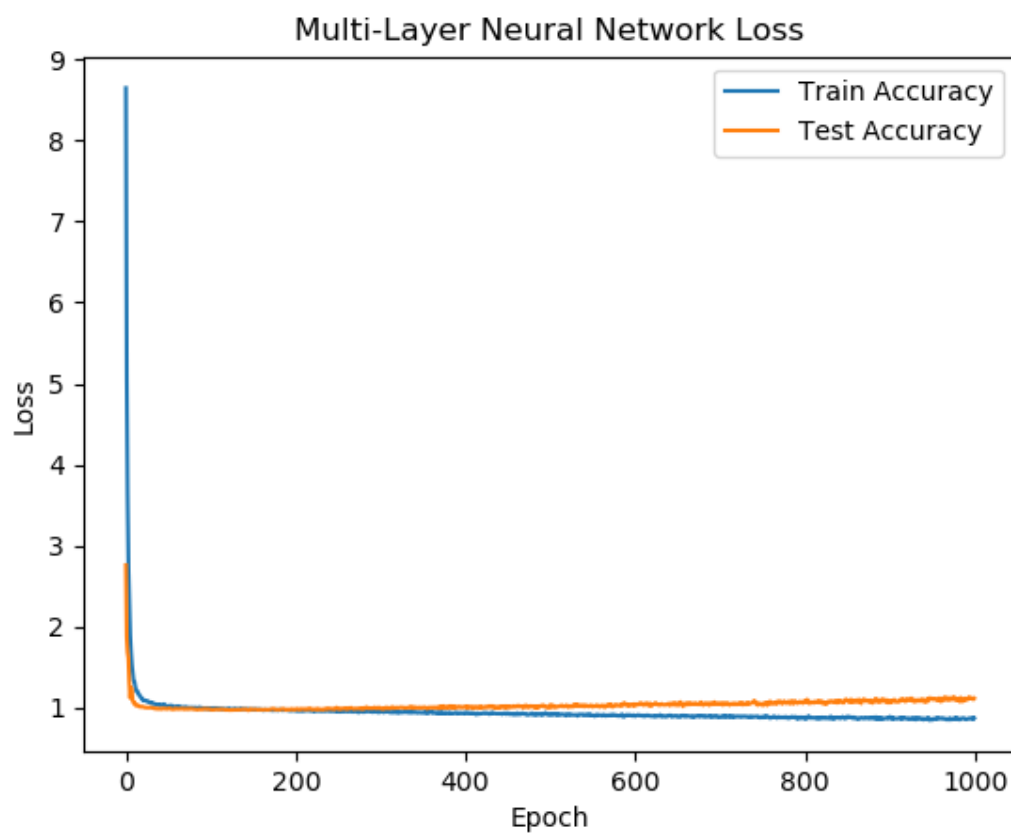


Εικόνα Γραφήματος 6

Εικόνα πορείας συνάρτησης σφάλματος:



Εικόνα Γραφήματος 7



Εικόνα Γραφήματος 8

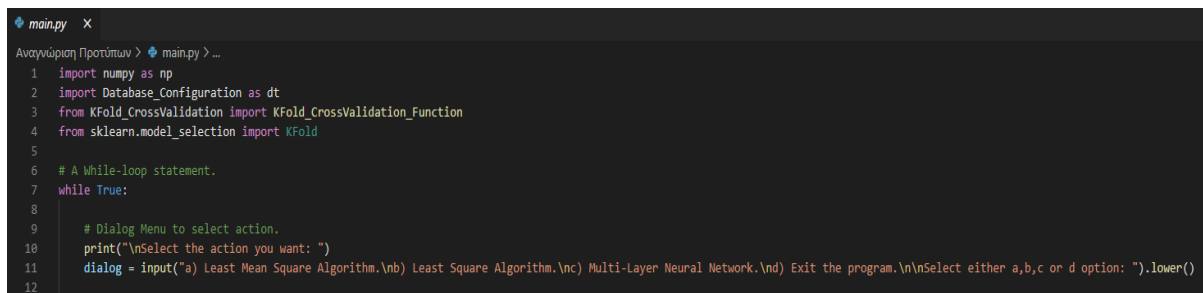
Κεντρικό Αρχείο Εκτέλεσης – main

Στη συγκεκριμένη ενότητα θα αναφερθούμε στο κεντρικό εκτελέσιμο αρχείο της εφαρμογής, το αρχείο **main.py**.

Με το κάλεσμα του συγκεκριμένου αρχείου ξεκινάει να εκτελείτε η παρούσα υλοποίησή.

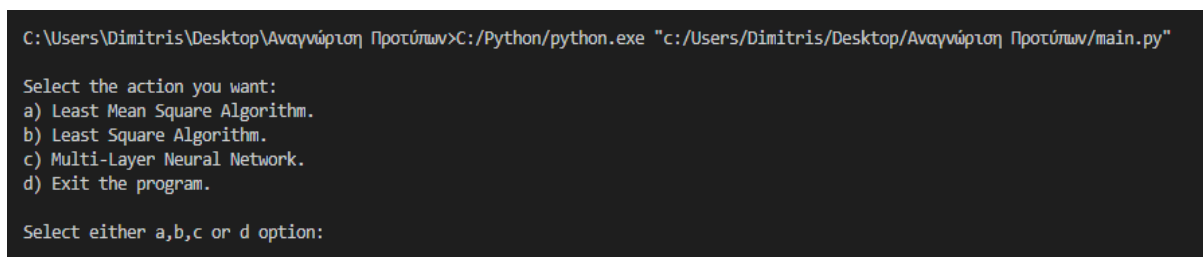
Αρχικά, εισάγουμε τις απαραίτητες βιβλιοθήκες και τα αρχεία **Database_Configuration** και **KFoldCrossValidation**.

Στη συνέχεια, εμφανίζουμε στο χρήστη μέσω της κονσόλας το μενού των επιλογών, ώστε να επιλέξει τον αλγόριθμο που επιθυμεί είτε να σταματήσει το πρόγραμμα.



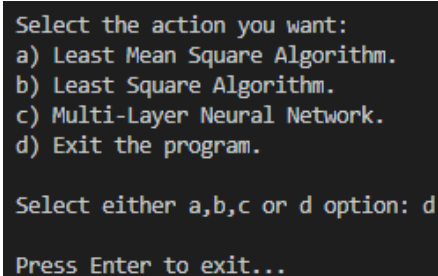
```
main.py X
Αναγνώριση Προτύπων > main.py > ...
1 import numpy as np
2 import Database_Configuration as dt
3 from KFold_CrossValidation import KFold_CrossValidation_Function
4 from sklearn.model_selection import KFold
5
6 # A While-loop statement.
7 while True:
8
9     # Dialog Menu to select action.
10    print("\nSelect the action you want: ")
11    dialog = input("a) Least Mean Square Algorithm.\nb) Least Square Algorithm.\nc) Multi-Layer Neural Network.\nd) Exit the program.\n\nSelect either a,b,c or d option: ").lower()
12
```

Εικόνα Εφαρμογής 35



```
C:\Users\Dimitris\Desktop\Αναγνώριση Προτύπων>C:/Python/python.exe "C:/Users/Dimitris/Desktop/Αναγνώριση Προτύπων/main.py"
Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.
Select either a,b,c or d option: d
```

Εικόνα Εφαρμογής 36



```
Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.
Select either a,b,c or d option: d
Press Enter to exit...
```

Εικόνα Εφαρμογής 37

Εάν η επιλογή του χρήστη είναι έγκυρη, τότε το πρόγραμμα συνδέεται με το αρχείο του KFold και ξεκινάει τη διαβίβαση των δεδομένων για την επίλυση του προβλήματος (τα δεδομένα που έλαβε από το αρχείο της Βάσης).

```
C:\Users\Dimitris\Desktop\Αναγνώριση Προτύπων>C:/Python/python.exe "c:/Users/Dimitris/Desktop/Αναγνώριση Προτύπων/main.py"

Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.

Select either a,b,c or d option: e

Invalid input, please try again.

Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.

Select either a,b,c or d option: █
```

Εικόνα Εφαρμογής 38

Σε αντίθετη περίπτωση, θα εμφανίζεται σχετικό μήνυμα σφάλματος και το μενού των επιλογών θα ξαναεμφανίζεται.

```
main.py ×
Αναγνώριση Προτύπων > main.py > ...

13     # If user's response belongs to the
14     # aforementioned responses range.
15     if dialog in ["a", "b", "c", "d"]:
16
17         # Response's code for a-option, Least Mean Squares Algorithm.
18         if dialog == "a":
19
20             # Initialize important variables, in order to utilize
21             # the Least Mean Squares Algorithm.
22             import LeastMeanSquares as lms
23
24             # Betting Odd's Table.
25             x = np.array(dt.K)
26             # Output - Label's Table.
27             y = np.array(dt.Y)
28
29             # Call the K-Fold Algorithm (10 Folds),
30             # with category equals to 1.
31             KFold_CrossValidation_Function(x, y, 1)
32
33         # Response's code for b-option, Least Squares Algorithm.
34         elif dialog == "b":
35
36             # Initialize important variables, in order to utilize
37             # the Least Squares Algorithm.
38             import LeastSquares as ls
39
40             # Betting Odd's Table.
41             x = np.array(dt.K)
42             # Output - Label's Table.
43             y = np.array(dt.Y)
44
45             # Call the K-Fold Algorithm (10 Folds),
46             # with category equals to 2.
47             KFold_CrossValidation_Function(x, y, 2)
```

Εικόνα Εφαρμογής 39

Έπειτα, το κεντρικό αρχείο εφόσον η επιλογή του χρήστη ολοκληρωθεί (και τα 10 Folds) θα ξαναεμφανίζει το μενού των επιλογών για πιθανή επανεκτέλεση κάποιου εκ των αλγορίθμων.

```
main.py X
Αναγνώριση Προτύπων > main.py > ...
49     # Response's code for c-option, Multi-Layer Neural Network.
50     elif dialog == "c":
51
52         # Initialize important variables, in order to utilize
53         # the Multi-Layer Neural Network.
54         import MultiLayerNN as mnn
55
56         # Getting the Multi-Layer Neural
57         # Network's data for the tables.
58         x,y=dt.MNN_data()
59
60         # Betting Odd's Table.
61         x = np.array(x)
62         # Output - Label's Table.
63         y = np.array(y)
64
65         # Call the K-Fold Algorithm (10 Folds),
66         # with category equals to 3.
67         KFold_CrossValidation_Function(x, y, 3)
68
69     # Response's code for d-option (exit the program).
70     elif dialog == "d":
71         input("\nPress Enter to exit...")
72         break
73
74     else:
75         # Console's message regarding Invalid Input Error.
76         print("\nInvalid input, please try again.")
77 else:
78     # Console's message regarding Invalid Input Error.
79     print("\nInvalid input, please try again.")
```

Εικόνα Εφαρμογής 40

```
C:\Users\Dimitris\Desktop\Αναγνώριση Προτύπων>C:/Python/python.exe "c:/Users/Dimitris/Desktop/Αναγνώριση Προτύπων/main.py"

Select the action you want:
a) Least Mean Square Algorithm.
b) Least Square Algorithm.
c) Multi-Layer Neural Network.
d) Exit the program.

Select either a,b,c or d option: b
```

Εικόνα Εφαρμογής 41

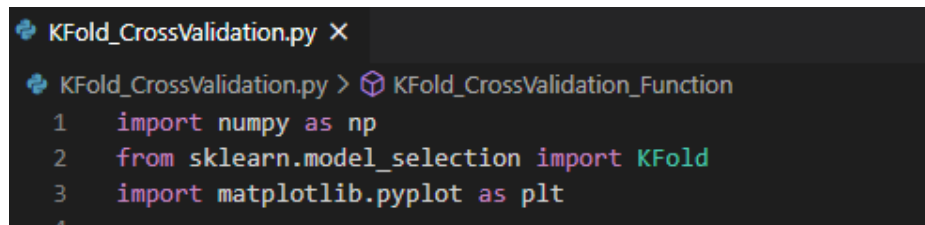
K-Fold Cross Validation Algorithm

Στη παρούσα ενότητα θα γίνει εκτενής αναφορά για την μέθοδο της 10-πλής Διεπικύρωσης (10 Fold Cross Validation Algorithm).

Ειδικότερα, με το συγκεκριμένο αρχείο υλοποιείται η διαχώριση των δεδομένων σε 10 πακέτα (folds/partitions) από τα οποία τα 9 θα τα χρησιμοποιήσουμε για την εκπαίδευση (training) και το άλλο (το δέκατο) για την επικύρωση – τεστ (validation).

Ωστόσο, σε κάθε επανάληψη (iteration) αλλάζουν τα πακέτα προκειμένου να φανερωθεί ποιος συνδυασμός έχει τα καλύτερα αποτελέσματα.

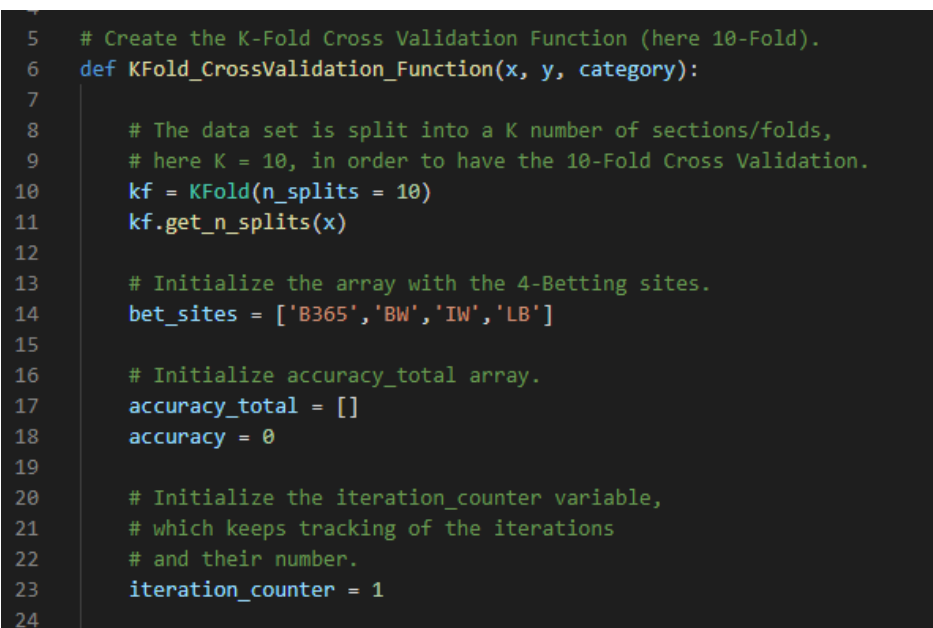
Για να επιτευχθούν στην ολότητά τους τα παραπάνω χρειάζεται αρχικά να εισάγουμε τις παρακάτω απεικονιζόμενες εικόνες.



```
KFold_CrossValidation.py X
KFold_CrossValidation.py > KFold_CrossValidation_Function
1 import numpy as np
2 from sklearn.model_selection import KFold
3 import matplotlib.pyplot as plt
4
```

Εικόνα Εφαρμογής 42

Στη συνέχεια και μέσα από τη συνάρτηση **KFold_CrossValidation**, διαχωρίζουμε τη βάση δεδομένων σε 10 πακέτα – folds και αρχικοποιούμε τις απαραίτητες μεταβλητές, μεταξύ των οποίων και ο μετρητής των επαναλήψεων (*iteration_counter*).



```
5 # Create the K-Fold Cross Validation Function (here 10-Fold).
6 def KFold_CrossValidation_Function(x, y, category):
7
8     # The data set is split into a K number of sections/folds,
9     # here K = 10, in order to have the 10-Fold Cross Validation.
10    kf = KFold(n_splits = 10)
11    kf.get_n_splits(x)
12
13    # Initialize the array with the 4-Betting sites.
14    bet_sites = ['B365', 'BW', 'IW', 'LB']
15
16    # Initialize accuracy_total array.
17    accuracy_total = []
18    accuracy = 0
19
20    # Initialize the iteration_counter variable,
21    # which keeps tracking of the iterations
22    # and their number.
23    iteration_counter = 1
24
```

Εικόνα Εφαρμογής 43

Έπειτα, και μέσα από μια επανάληψη `for` εκτυπώνονται τα κατάλληλα μηνύματα στο χρήστη σχετικά με τον αριθμό της επανάληψης και την έναρξη της διαδικασίας της εκπαίδευσης.

Επιπλέον, στο συγκεκριμένο σημείο φορτώνονται τα δεδομένα από την `main` (η οποία τα δέχεται από την `Database_Configuration` και τα τροποποιεί κατάλληλα).

```
25 # A For-Loop Statement, in order to get each of the 10 folds/partitions
26 # and present its results, depending on the category - user's choice.
27 for train_index, test_index in kf.split(x):
28
29     # Console's Message, regarding the iteration number, appears.
30     print('\nProcess: Iteration No: ' + str(iteration_counter))
31
32     # Console's Message, regarding the training process, appears.
33     print('Please wait. The Training Process has been started!')
34
35     # Training and Testing of the Betting Odd's Table.
36     x_train, x_test = x[train_index], x[test_index]
37
38     # Training and Testing of the Output - Label's Table.
39     y_train, y_test = y[train_index], y[test_index]
```

Εικόνα Εφαρμογής 44

Εάν ο χρήστης επιλέξει την πρώτη κατηγορία, δηλαδή τον αλγόριθμο Least Mean Square, τότε θα ακολουθείτε η απεικονιζόμενη διαδικασία:

```
41 # If user select the first (a) option,
42 # the Least Mean Squares Algorithm.
43 if(category == 1):
44
45     # Initialize the theta array.
46     theta=[1,1,1,1]
47
48     # Initialize important variables, in order to utilize
49     # the Least Mean Squares Algorithm.
50     import LeastMeanSquares as lms
51     matrix_weights, performance, output = lms.init(theta, x_train, y_train)
52
53     # Finds the Predicted Matches' Accuracy by the Algorithm.
54     accuracy = lms.test(matrix_weights[np.argmin(performance)], x_test, y_test, output)
55
56     # A For-Loop, in order to find the Most Accurate Betting site.
57     for k in bet_sites:
58         if (bet_sites.index(k) == np.argmin(performance)):
59             most_accurate_betting_site = k
60             break
61
62     plot_colors = ['green','blue','red','brown']
63     plt.bar(['B365','IW','BW','LB'], performance, color = plot_colors, width = 0.3)
64     plt.title('Results for Iteration No: ' + str(iteration_counter), fontsize=14)
65     plt.xlabel('Betting Sites', fontsize=14)
66     plt.ylabel("Prediction's Accuracy", fontsize=14)
67     plt.ylim([0.65, 0.78])
68     plt.grid(False)
69     plt.show()
```

Εικόνα Εφαρμογής 45

Ειδικότερα, αρχικοποιούνται και καλούνται οι απαραίτητες συναρτήσεις από το αρχείο του αλγορίθμου που επιλέχθηκε από το χρήστη (*Least Mean Square*), αναδεικνύεται η καλύτερη στοιχηματική και δημιουργούνται τα δύο γραφήματα που προαναφέρθηκαν.

```
70
71     # If iteration_counter equals to 10.
72     if (iteration_counter == 10):
73         accuracy_total.append(accuracy)
74
75         # On the 10th iteration print the
76         # total accuracy plot.
77         # Create the plot.
78         plt.figure(figsize = (8, 6), dpi=80)
79         plt.subplot(1, 1, 1)
80         plt.plot([1,2,3,4,5,6,7,8,9,10], accuracy_total, color='orange', linewidth=4.0, linestyle='solid')
81         plt.title('Least Mean Square Algorithm - Total Accuracy Results', fontsize=14)
82         plt.xlabel('Iterations', fontsize=14)
83         plt.ylabel("Model's Accuracy", fontsize=14)
84         plt.legend(["Model's Accuracy"], loc = 'upper left')
85         plt.ylim(60, 85)
86         plt.grid(False)
87         plt.show()
88
89     # iteration_counter != 10.
90     else:
91         accuracy_total.append(accuracy)
92
93     # Console's Message, regarding the Most Accurate Betting Site appears.
94     print('Most Accurate Betting Site: ' + str(most_accurate_betting_site))
95
96     # Console's Message, regarding Algorithm Weights appears.
97     print('Weights: ' + str(matrix_weights[np.argmin(performance)]))
98
99     # Console's Message, regarding the iteration number, appears.
100    print('Completed: Iteration No: ' + str(iteration_counter))
```

Εικόνα Εφαρμογής 46

Αντίστοιχη είναι η διαδικασία που θα ακολουθήσει το πρόγραμμα εάν ο χρήστης επιλέξει τον δεύτερο αλγόριθμο, Least Square.

```
102 # Else If user select the second (b) option,  
103 # the Least Squares Algorithm.  
104 elif(category==2):  
105  
106     # Initialize the theta array.  
107     theta=[1,1,1]  
108  
109     # Initialize important variables, in order to utilize  
110     # the Least Squares Algorithm.  
111     import LeastSquares as ls  
112     accuracy = ls.main(x_train,y_train,x_test,y_test)  
113     performance = ls.init(theta, x_train, y_train)  
114  
115     # A For-Loop, in order to find the Most Accurate Betting site.  
116     for k in bet_sites:  
117         if (bet_sites.index(k) == np.argmin(performance)):  
118             most_accurate_betting_site = k  
119             break  
120  
121     # If iteration_counter equals to 10.  
122     if (iteration_counter == 10):  
123         accuracy_total.append(accuracy)  
124  
125         # On the 10th iteration print the  
126         # total accuracy plot.  
127         # Create the plot.  
128         plt.figure(figsize = (8, 6), dpi=80)  
129         plt.subplot(1, 1, 1)  
130         plt.plot([1,2,3,4,5,6,7,8,9,10], accuracy_total, color='cyan', linewidth=4.0, linestyle='solid')  
131         plt.title('Least Square Algorithm - Total Accuracy Results', fontsize=14)  
132         plt.xlabel('Iterations', fontsize=14)  
133         plt.ylabel("Model's Accuracy", fontsize=14)  
134         plt.legend(["Model's Accuracy"], loc = 'upper left')  
135         plt.ylim(40, 65)  
136         plt.grid(False)  
137         plt.show()  
138  
139     # iteration_counter != 10.  
140     else:  
141         accuracy_total.append(accuracy)  
142  
143     # Console's Message, regarding the Most Accurate Betting Site appears.  
144     print('Most Accurate Betting Site: ' + str(most_accurate_betting_site))  
145  
146     # Console's Message, regarding the iteration number, appears.  
147     print('Completed: Iteration No: ' + str(iteration_counter))
```

Εικόνα Εφαρμογής 47

Πιο συγκεκριμένα, αρχικοποιούνται και καλούνται οι απαραίτητες συναρτήσεις από το αρχείο του αλγορίθμου που επιλέχθηκε από το χρήστη (*Least Square*), αναδεικνύεται η καλύτερη στοιχηματική και δημιουργούνται τα δύο γραφήματα που προαναφέρθηκαν.

Τέλος, εάν ο χρήστης επιλέξει την επιλογή του Πολυστρωματικού Νευρωνικού Δικτύου, τότε μέσω του αρχείου *KFold_CrossValidation*, ανά επανάληψη – fold θα εισάγονται οι απαραίτητες μεταβλητές από τις μεθόδους του αρχείου του Νευρωνικού (*MultiLayerNN*) και θα ξεκινάει η εκτέλεση του εν λόγω αλγορίθμου.

```
149     # Else If user select the third (c) option,  
150     # the Multi-Layer Neural Network.  
151     else:  
152         # Initialize important variables, in order to utilize  
153         # the Multi-Layer Neural Network.  
154         import MultiLayerNN as mnn  
155         mnn.init(x_train,y_train,x_test,y_test)  
156  
157  
158     # Increase the iteration_counter, in order to get the next iteration.  
159     iteration_counter += 1
```

Εικόνα Εφαρμογής 48

Αξίζει να σημειωθεί το γεγονός ότι σε αντίθεση με τα δύο προηγούμενα ερωτήματα, στο τρίτο τα γραφήματα δημιουργούνται στο αρχείο του αλγορίθμου και όχι σε αυτό του K – Fold.

Αυτό επιλέχθηκε καθώς, η συγκεκριμένη υλοποίηση γίνεται με τη βοήθεια της βιβλιοθήκης **keras** και η δημιουργία των γραφημάτων διευκολύνθηκε αρκετά με την παραπάνω συνδυαστική υλοποίηση, ενώ τα αποτελέσματα είναι ικανοποιητικά.

Οδηγίες Εκτέλεσης Προγράμματος

Στη συγκεκριμένη ενότητα θα αναφερθούμε στο τρόπο με τον οποίο θα εκτελεστεί ορθολογικά το εκτελέσιμο, που δημιουργήσαμε.

Ως εκ τούτου, για να εκτελέσουμε τον κώδικα ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε τη γραμμή εντολών και μεταβαίνουμε στον αντίστοιχο φάκελο: \Αναγνώριση Προτύπων.
2. Πληκτρολογούμε το όνομα του αρχείου **main.py**, ώστε να ανοίξει το κεντρικό αρχείο του εκτελέσιμου.

Το αρχείο εκτελείται επιτυχώς και τα αποτελέσματα εμφανίζονται όπως έχει προαναφερθεί στις παραπάνω ενότητες και ανάλογα με το ερώτημα που θα επιλέξει ο χρήστης.

Βιβλιογραφία

Οι βιβλιογραφικές πηγές που χρησιμοποιήθηκαν στην εφαρμογή είναι οι ακόλουθες :

1. Διαχείριση Δεδομένων από την Kaggle
(<https://www.kaggle.com/docs/api>)
Τελευταία Προσπέλαση 08/01/2021
2. Shuffle Δεδομένων για καλύτερη ακρίβεια αποτελεσμάτων.
(<https://datascience.stackexchange.com/questions/24511/why-should-the-data-be-shuffled-for-machine-learning-tasks>)
Τελευταία Προσπέλαση 13/02/2021
3. Δημιουργία Πολυστρωματικού Νευρωνικού Δικτύου.
(<https://towardsdatascience.com/building-neural-network-from-scratch-9c88535bf8e9>)
Τελευταία Προσπέλαση 28/01/2021
4. Δημιουργία Γραφημάτων αναλόγως το ερώτημα.
Documentation βιβλιοθήκης matplotlib.
(https://matplotlib.org/stable/api/as_gen/matplotlib.pyplot.plot.html)
Τελευταία Προσπέλαση 17/02/2021
5. Δημιουργία Γραφημάτων τύπου Bar Chart (ερωτήματα 1 & 2 – γραφήματα για τις 4 στοιχηματικές)
(<https://datatofish.com/bar-chart-python-matplotlib/>)
Τελευταία Προσπέλαση 20/02/2021
6. Υλοποίηση γραφημάτων με πολλαπλά δεδομένα
(<https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed>)
Τελευταία Προσπέλαση 20/02/2021



7. Δημιουργία Γραφημάτων τύπου Ιστογράμματος (γραφήματα συνολικής ευστοχίας)

(<https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/>)

Τελευταία Προσπέλαση 25/02/2021

8. Υλοποίηση μέσω της tensorflow

(https://www.tensorflow.org/api_docs/python/tf/keras)

Τελευταία Προσπέλαση 15/02/2021

9. Υλοποίηση μέσω της keras

(<http://man.hubwiz.com/docset/Keras.docset/Contents/Resources/Documents/index.html>)

Τελευταία Προσπέλαση 16/02/2021

Περιεχόμενα Απεσταλμένου Αρχείου

Το τελικό αρχείο της εργασίας (Εργασία Αναγνώρισης Προτύπων.zip) θα περιέχει τα παρακάτω:

1. Τα αρχεία πηγαίου κώδικα σε γλώσσα Python που βρίσκονται στο φάκελο Αναγνώριση Προτύπων.
2. Το κείμενο τεκμηρίωσης, με τίτλο Εργασία Αναγνώρισης Προτύπων.pdf .
3. Το αρχείο συνοπτικής παρουσίασης, όπου θα εστιάζει στα κύρια σημεία της εργασίας, με τίτλο Παρουσίαση Αναγνώρισης Προτύπων.ppsx.
4. Το αρχείο με τα στιγμιότυπα της εφαρμογής, που χρησιμοποιήθηκαν για την τεκμηρίωση της υλοποίησης, με τίτλο Screenshots.
5. Το αρχείο με τη συνολική βιβλιογραφία της εφαρμογής, με όλες τις πηγές που χρησιμοποιήθηκαν για υλοποίησή της, με τίτλο Βιβλιογραφία.txt .