

Αναγνώριση ΠροτύπωνΑπαλλακτική Εργασία

*Ακαδημαϊκό Έτος 2020 - 2021*

Δημήτρης Ματσαγγάνης, Π17068

Περιεχόμενα

[Εισαγωγή Εργασίας 3](#_Toc63797967)

[Κεντρική Ιδέα Υλοποίησης 5](#_Toc63797968)

[Μορφοποίηση Παρεχόμενης Βάσης Δεδομένων 6](#_Toc63797969)

[Θέμα 1ο: Αλγόριθμος Ελάχιστου Μέσου Τετραγωνικού Σφάλματος - Least Mean Squares 12](#_Toc63797970)

## 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 και το αντίστοιχο σύνολο των διαφορετικών ομάδων ως , τότε η πιο αφηρημένη αναπαράσταση του κάθε αγώνα μπορεί να πραγματοποιηθεί ως μια διατεταγμένη τριάδα της μορφής , με , όπου η είναι η ομάδα που αγωνίζεται εντός έδρας (home team), είναι η ομάδα που αγωνίζεται εκτός έδρας (away team) και 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 **Α** (away win) υποδηλώνει την νίκη της ομάδας που αγωνίζεται εκτός έδρας. Αν με και συμβολίσουμε το πλήθος των τερμάτων που επιτυγχάνονται από τις ομάδες εντός και εκτός έδρας κατά την διεξαγωγή του αγώνα , τότε η μεταβλητή έκβασης του αγώνα μπορεί να ορισθεί συναρτήσει της διαφοράς των συνολικών τερμάτων ως:

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 είναι συσχετισμένη με ένα διάνυσμα χαρακτηριστικών τα οποία αντιστοιχούν στις παρακάτω στήλες του πίνακα **Team\_Attributes** {**buildUpPlaySpeed**, **buildUpPlayPassing**, **chanceCreationPassing**, **chanceCreationCrossing**, **chanceCreationShooting**, **defencePressure**, **defenceAggregation**, **defenceTeamWidth**}. Επιπλέον, κάθε αγώνας είναι συσχετισμένος με τέσσερα διανύσματα προγνωστικών με τα αντιστοιχούν στις παρακάτω στήλες του πίνακα **Match** {**B365H**, **B365D**, **B365A**,**BWH**, **BWD**, **BWA**, **IWH**, **IWD**, **IWA**, **LBH**, **LBD**, **LBA**}. Δηλαδή, το κάθε διάνυσμα συγκεντρώνει τις στοιχηματικές αποδόσεις για κάθε πιθανή έκβαση του αγώνα για κάθε στοιχηματική εταιρεία . 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:**

1. Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Μέσου Τετραγωνικού Σφάλματος (**Least Mean Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφήςγια κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.
2. Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Τετραγωνικού Σφάλματος (**Least Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφήςγια κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.
3. Να υλοποιήσετε ένα πολυστρωματικό νευρωνικό δίκτυο, ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί μια συνάρτηση διάκρισης της μορφής  **,** όπου το αντιστοιχεί στο πλήρες διάνυσμα χαρακτηριστικών του κάθε αγώνα που δίνεται από την σχέση:

**Παρατηρήσεις:**

1. 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).**
2. 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.
3. Παραδοτέα της εργασίας αποτελούν ο **κώδικας** της υλοποίησής σας σε MATLAB ή Python καθώς και ένα συνοδευτικό **κείμενο τεκμηρίωσης**.
4. Μπορείτε να εργασθείτε σε ομάδες των **δύο** ή **τριών** **ατόμων**.
5. 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/hugomathien/soccer) μέσω μιας βάσης δεδομένων sqlite (συγκεκριμένα την database.sqlite).

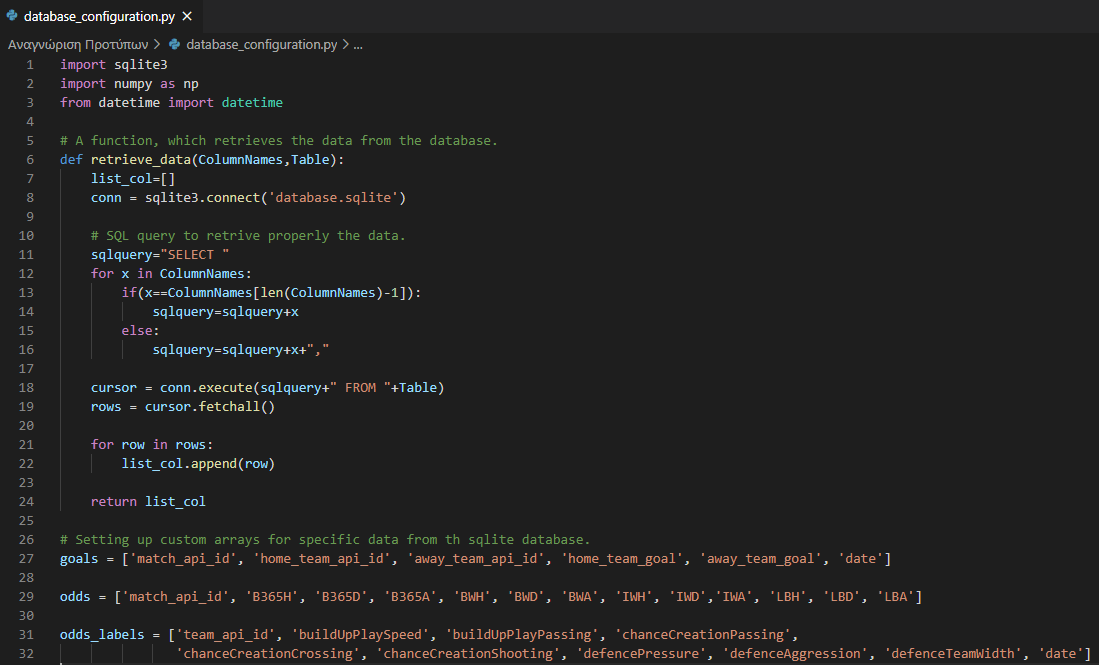
Για την ορθολογική απάντηση των τριών ερωτημάτων της άσκησης θα δημιουργήσουμε τρεις διαφορετικούς ταξινομητές ώστε να συγκρίνουμε τα αποτελέσματά τους, ακολουθώντας τις υποδείξεις της εκφώνησής της άσκησης.

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**.

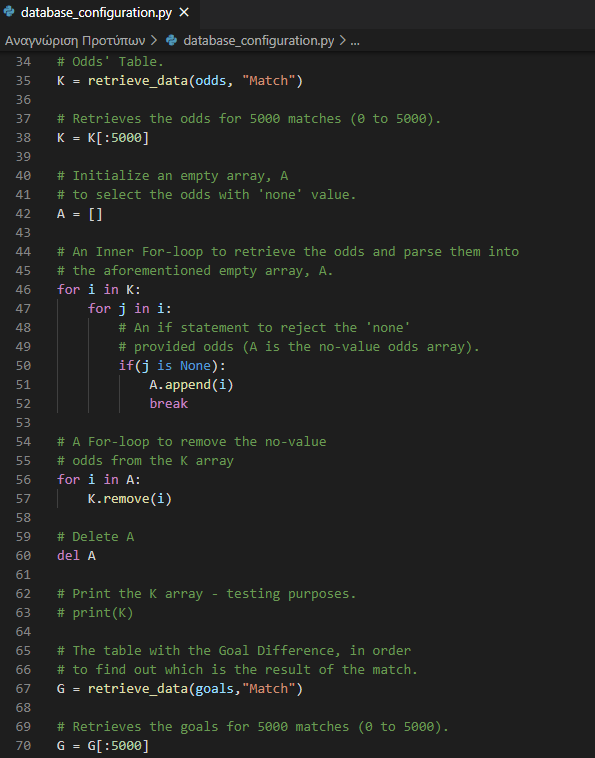


Εικόνα Εφαρμογής 1

H 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 τρείς custom λίστες, οι οποίες χειρίζονται και τροποποιούν τα δεδομένα κατάλληλα για τις συναρτήσεις που θα ακολουθήσουν.

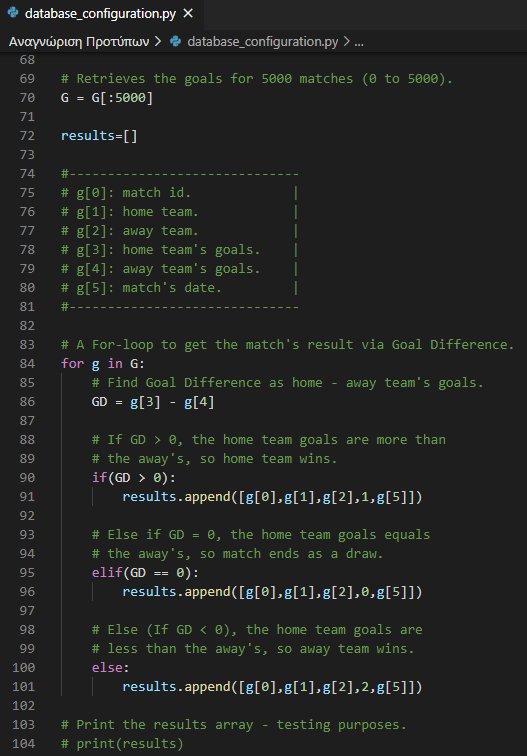
Πιο συγκεκριμένα, η πρώτη custom λίστα, Κ, που δημιουργείται αφορά τις στοιχηματικές αποδόσεις που δίνονται από τις εκάστοτε εταιρίες .



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.

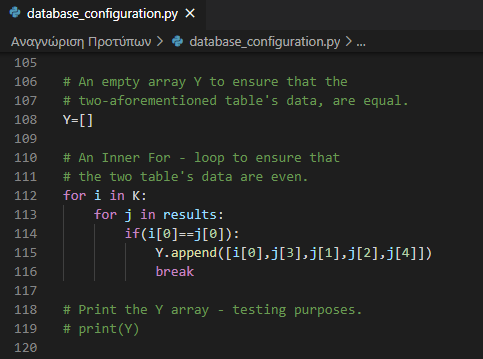
Η δεύτερη εξ αυτών, αφορά τα γκολ που σημειώθηκαν σε κάποιον αγώνα (ο οποίος θα βρίσκεται από το id του, *match\_api\_id*). Μέσα από αυτή τη διαδικασία βρίσκουμε την διαφορά των γκολ και το τελικό αποτέλεσμα του αγώνα, ανάλογα με τη διαφορά λοιπόν και το πρόσημο αυτής βρίσκουμε το αγώνα.



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”.

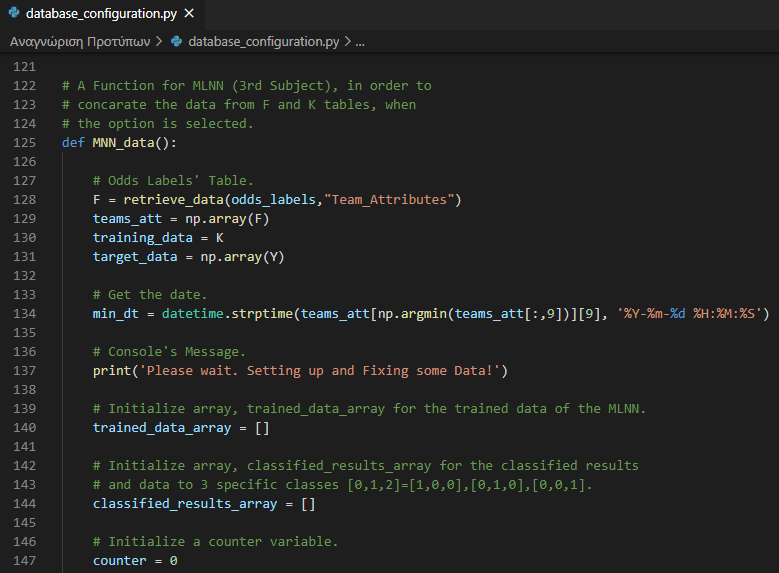
Τέλος, η τρίτη και τελευταία λίστα που δημιουργείται, έχει να κάνει με τις τελικές ετικέτες – labels που θα λαμβάνει κάθε απόδοση.



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).



Εικόνα Εφαρμογής 5

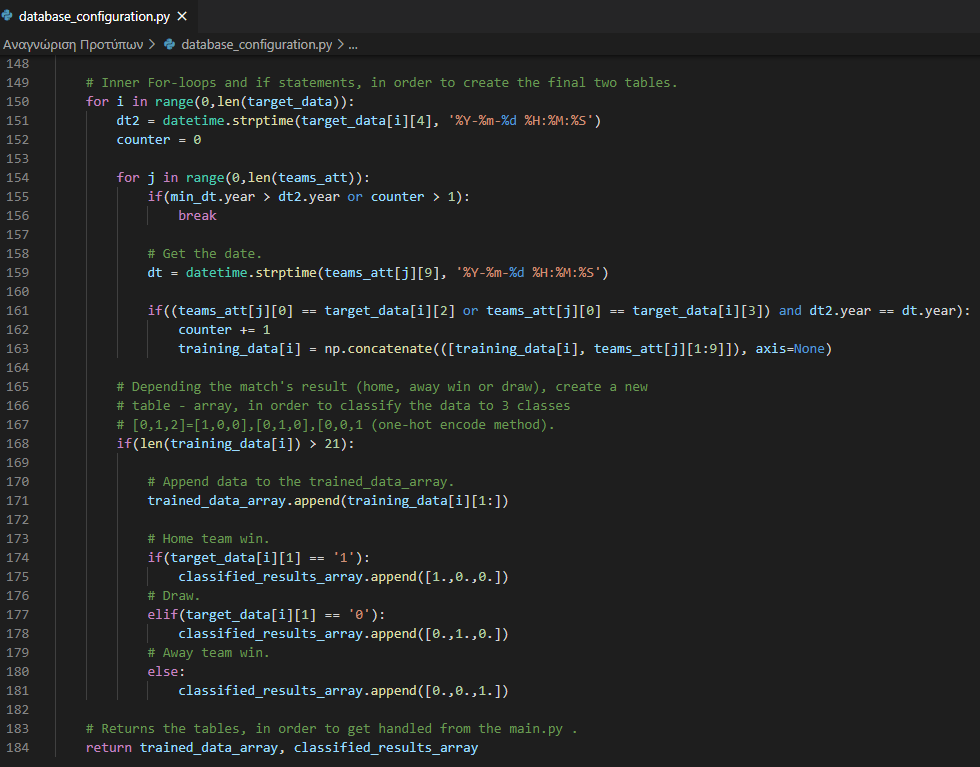
Η συνάρτηση **MNN\_data** καλείται κατά την επιλογή της υλοποίησης για το τρίτο ζητούμενο της άσκησης από την main.

Η συγκεκριμένη συνάρτηση στόχο έχει να κάνει “concarate” 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]



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)+ρ\*e\*x. (2), με ρ = learning rate e = y – yi = error.

Στη παρούσα υλοποίηση χρησιμοποιήθηκε η μέθοδος – λογική One Against All, καθώς ο αλγόριθμος Least Mean Squares είναι ένας δυαδικός 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 τρείς κατηγορίες που προαναφέρθηκαν είναι τα τρία πιθανά αποτελέσματα ενός ποδοσφαιρικού αγώνα, δηλαδή νίκη της γηπεδούχου ομάδας, ισοπαλία ή νίκη της φιλοξενούμενης ομάδας.

Ωστόσο, επειδή δεν μπορούμε να τις ταξινομήσουμε ταυτόχρονα και τις τρεις κατηγορίες θέτουμε ως άσσο 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.

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.

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 learn\_theta(), η οποία θα τρέξει τον τύπο της συνάρτησης μέσω της learnThetaSingle() αφού πρώτα προσθέσει τον άσσο στο τέλος κάθε στοιχηματικής απόδοσης (η μεταβλητή Χ) προκειμένου να βρεθεί το βάρος w0, μέχρι να γίνει το τελικό κόστος ελάχιστο.

Το τελικό κόστος υπολογίζεται μέσω της συνάρτησης costFunctinon(), η οποία υπολογίζει τον τύπο (Σ w\*x.T-y)^2/len(x).

Η συνάρτηση **hypothesis**() χρησιμοποιεί ως μεταβλητές τις αποδόσεις των στοιχηματικών εταιριών και τα labels από την learn\_theta(). 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.

Η συνάρτηση test(), η οποία 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.

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.

Δυστυχώς η διαδικασία εύρεσης του ελάχιστου κόστος 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 η ΒΕΤ365.

## Θέμα 2: Αλγόριθμος Ελάχιστου Τετραγωνικού Σφάλματος - Least Squares

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

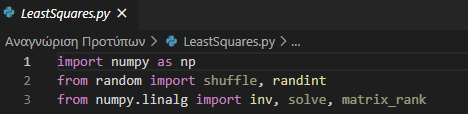
Ειδικότερα στο εν λόγο αρχείο κώδικα (αρχείο LeastSquares.py), υλοποιείται η λειτουργία του αλγορίθμου στη συγκεκριμένη περίπτωση αυτό γίνεται εφικτό με τη δημιουργία 5 μεθόδων – συναρτήσεων: train, test, predict, fixLabels και main.

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 o αλγόριθμος που υλοποιείται να έχει τη δυνατότητα να κατανοεί καλύτερα πώς τα δεδομένα που εισάγουμε σχετίζονται με τις κλάσεις (επιβλεπόμενη μάθηση - supervised learning).

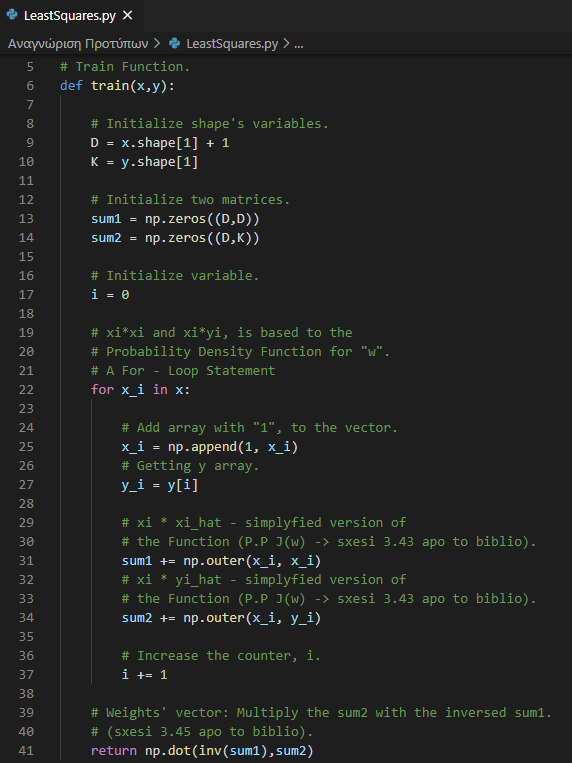
Πρωταρχικό βήμα είναι εισαγάγουμε τις απαραίτητες βιβλιοθήκες. Για τις μαθηματικές πράξεις μεταξύ πινάκων χρησιμοποιούμε τη βιβλιοθήκη **numpy**, για το ανακάτεμα των σειρών ενός πίνακα χρησιμοποιούμε τη **shuffle** και τη **randint,** από τη βιβλιοθήκη **random**. Ακολουθεί σχετική εικόνα από τον κώδικα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



Εικόνα Εφαρμογής 7

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

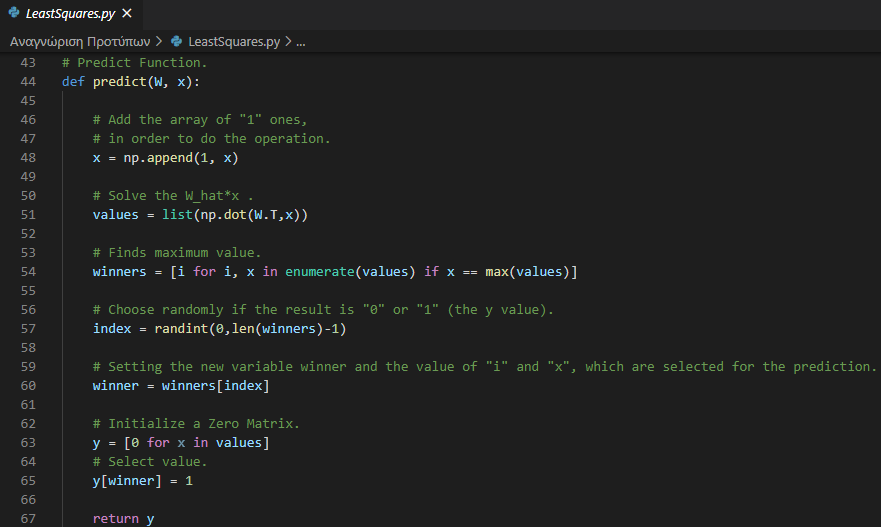
Καταρχάς στη συνάρτηση **train**, το **D** είναι το *πλήθος των χαρακτηριστικών*, ενώ το **K** είναι το *πλήθος των κλάσεων*. Έπειτα, αρχικοποιούμε τους πίνακες **sum1**, **sum2** και διαμορφώνουμε το διάνυσμα **x\_i**, ώστε να είναι έτοιμο για τις πράξεις που θα ακολουθήσουν. Επιλύουμε ως προς το διάνυσμα 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**.



Εικόνα Εφαρμογής 8

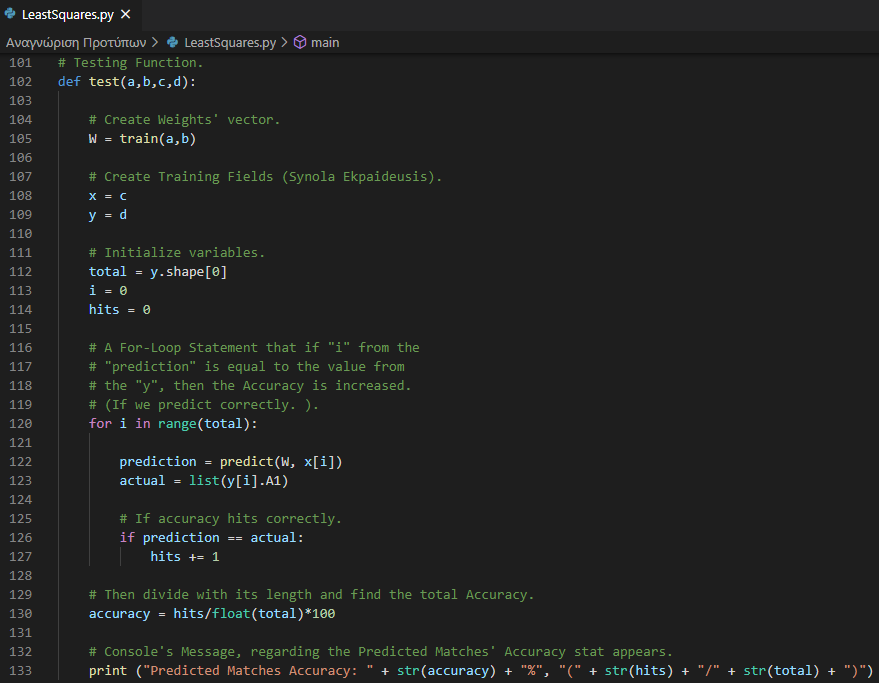
Ακόμη καλούμε την **train** για να διαχωρίσουμε τα δεδομένα μας στα 2 υποσύνολα, εκπαίδευσης και δοκιμής, κατασκευάζουμε μια δομή επανάληψης *for* και καλούμε την **predict** για κάθε κλάση.

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



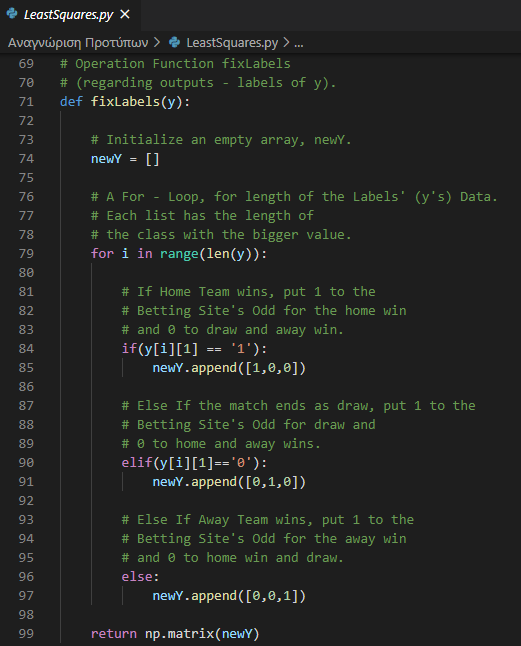
Εικόνα Εφαρμογής 9

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



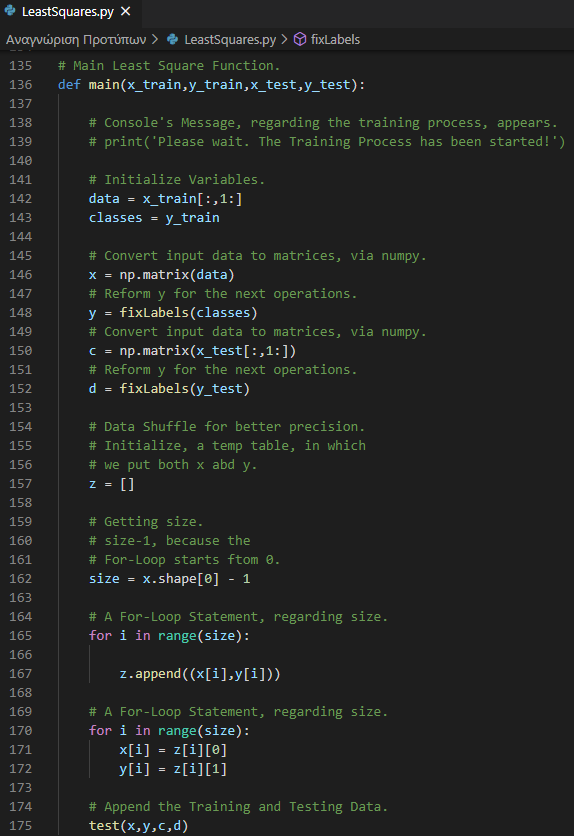
Εικόνα Εφαρμογής 10

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



Εικόνα Εφαρμογής 11

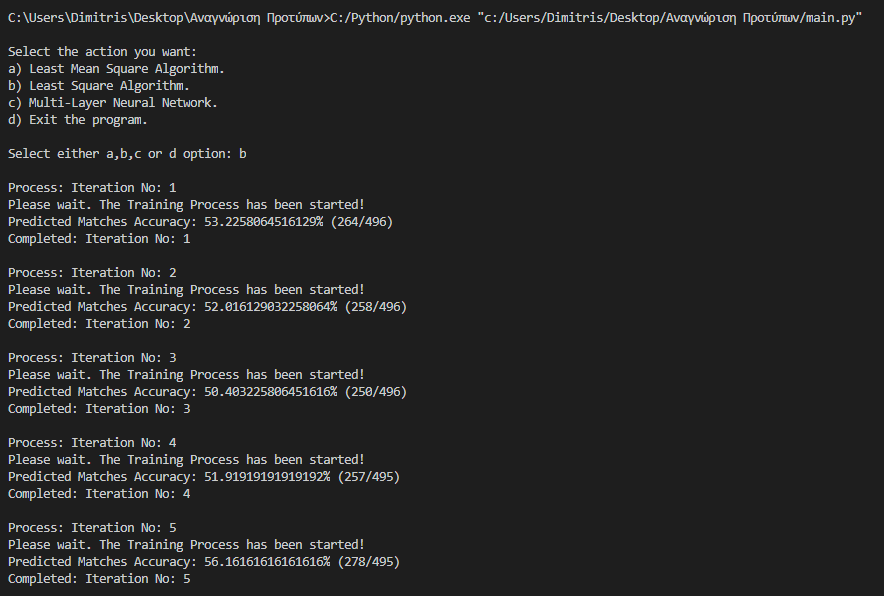
Τελικά, καλούμε τη πέμπτη κα τελευταία μέθοδο τη **main**, η οποία αρχικά εμφανίζει ένα μήνυμα έναρξης εκπαίδευσης μέσω της κονσόλας προς το χρήστη, παρέχει στους data και classes τα δεδομένα και τις κλάσεις αντίστοιχα. Στη συνέχεια τους μετατρέπουμε σε πίνακες x, y, μέσω της βιβλιοθήκης **numpy** και καλούμε τη **fixLabels** για τις μορφοποιήσεις στις κλάσεις μας, ενώ στο τέλος καλούμε την **test** και ξεκινάει η εκτέλεση του αλγορίθμου Ελάχιστου Τετραγωνικού Σφάλματος.



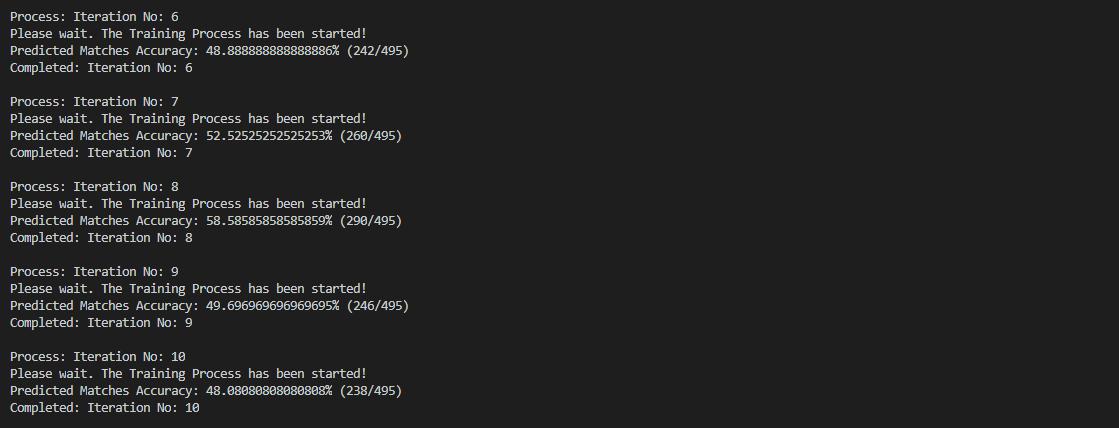
Εικόνα Εφαρμογής 12

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

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



Εικόνα Εφαρμογής 13



Εικόνα Εφαρμογής 14

## Θέμα 3: Πολυστρωματικό Νευρωνικό Δίκτυο – MultiLayer Neutral Network

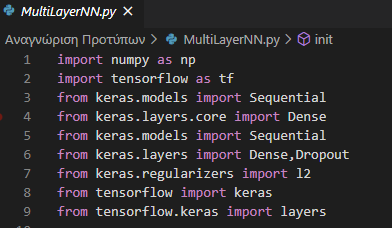
Στο τελευταίο ερώτημα της παρούσας υλοποίησης καλούμαστε να υλοποιήσετε ένα Πολυστρωματικό Νευρωνικό Δίκτυο (Multi-Layer Neutral Network).

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*](#_Μορφοποίηση_Παρεχόμενης_Βάσης)*)* 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.

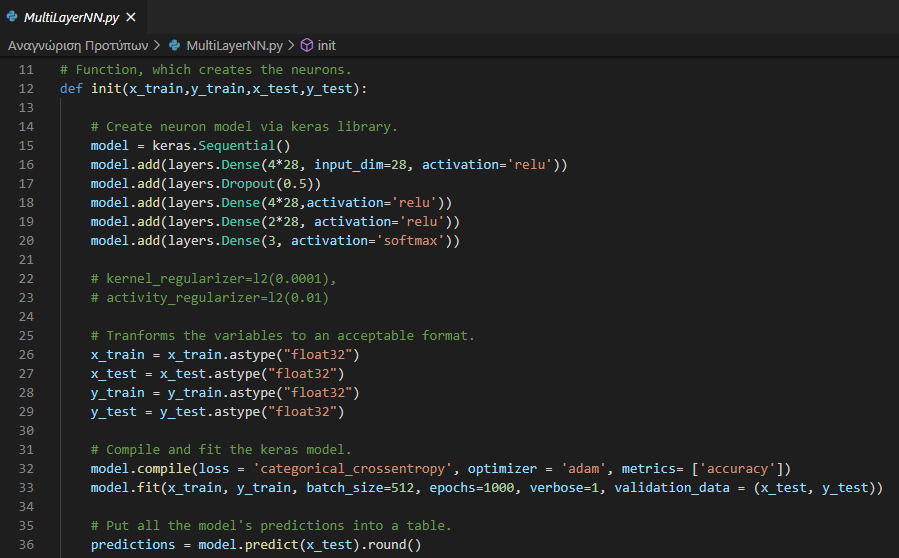
Για να γίνει αυτό πρέπει να εισάγουμε τις παρακάτω βιβλιοθήκες τις Python:



Εικόνα Εφαρμογής 15

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.



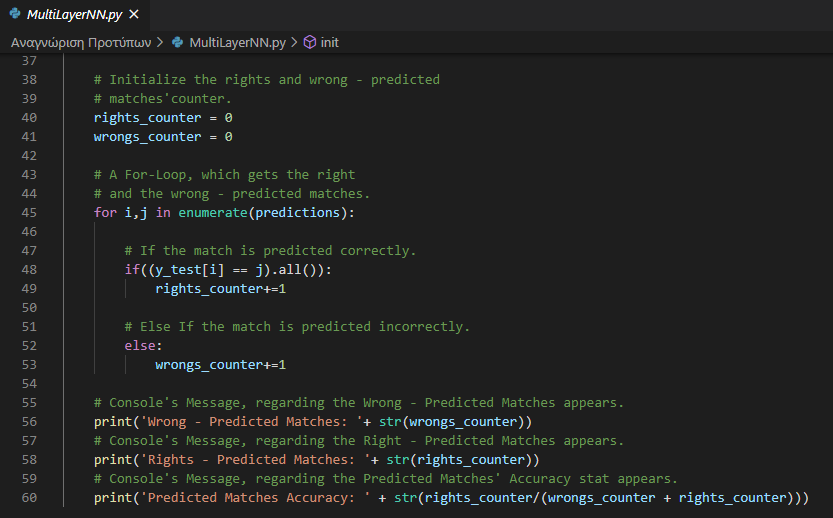
Εικόνα Εφαρμογής 16

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.

Kάνοντας 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 λάθος και τα εμφανίζει αφού πρώτα τα έχει κρατήσει σε πίνακα τις προβλέψεις που έκανε.



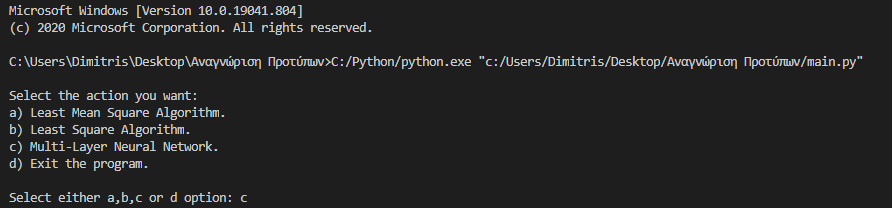
Εικόνα Εφαρμογής 17

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.

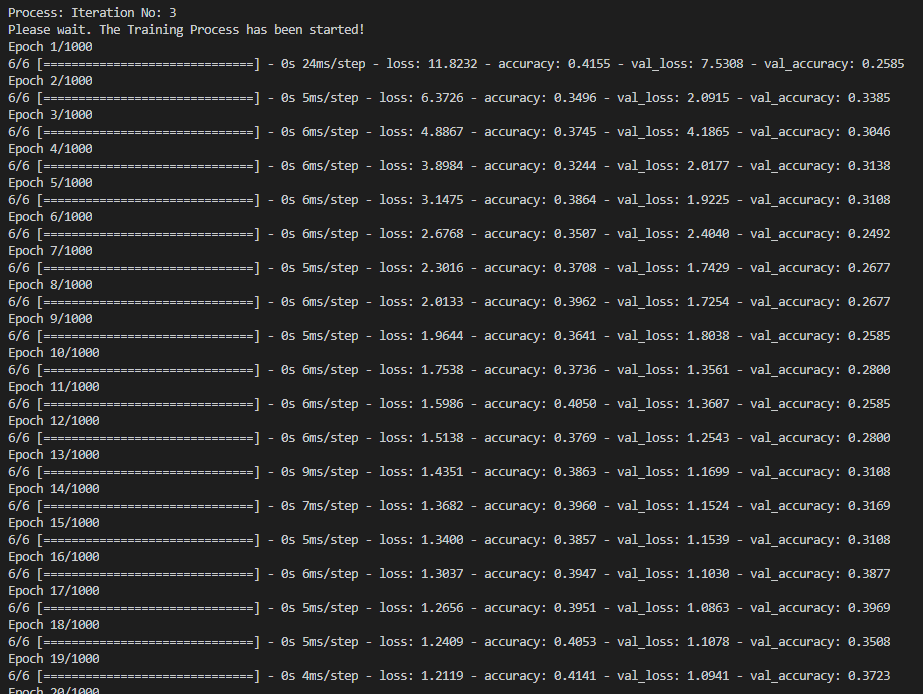
Ωστόσο δεν μπορούμε να σταματήσουμε το πρόγραμμα και την εκτέλεσή του εξαιτίας της χαμηλή εκπαίδευση πάλι θα έχουμε ένα παρόμοιο αποτέλεσμα.

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

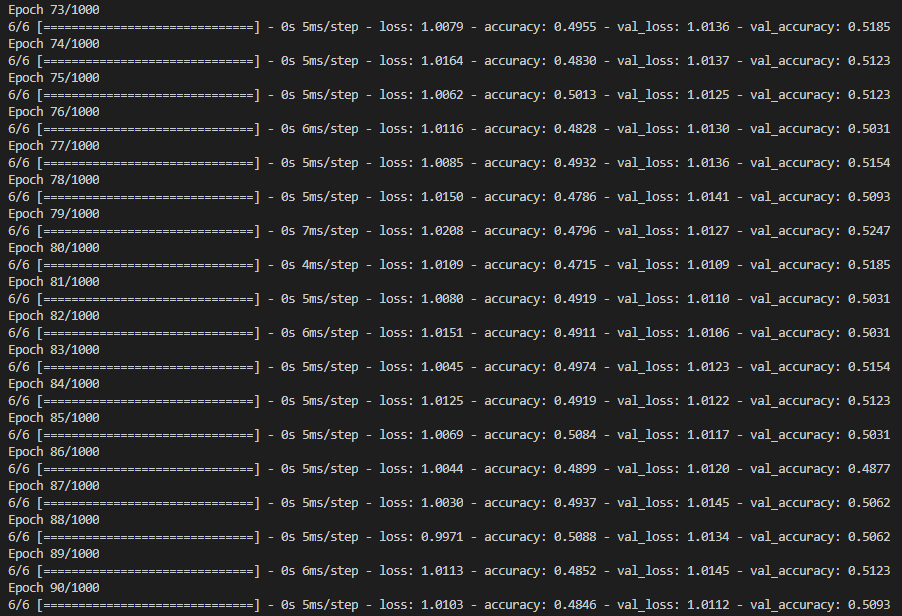
Ακολουθούν ενδεικτικά στιγμιότυπα από την εκτέλεση του προγράμματος :



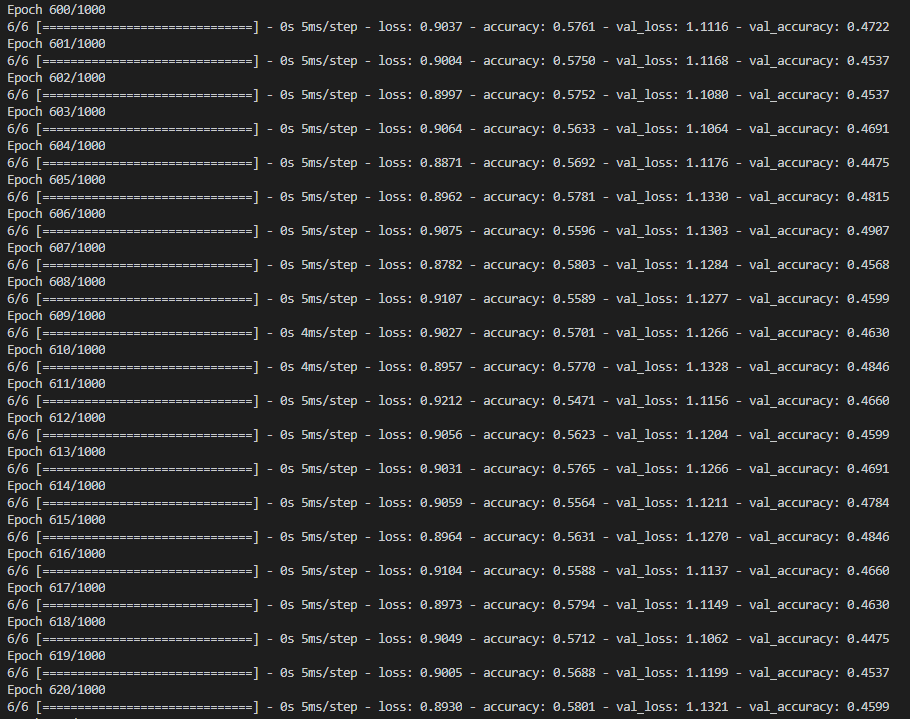
Εικόνα Εφαρμογής 18



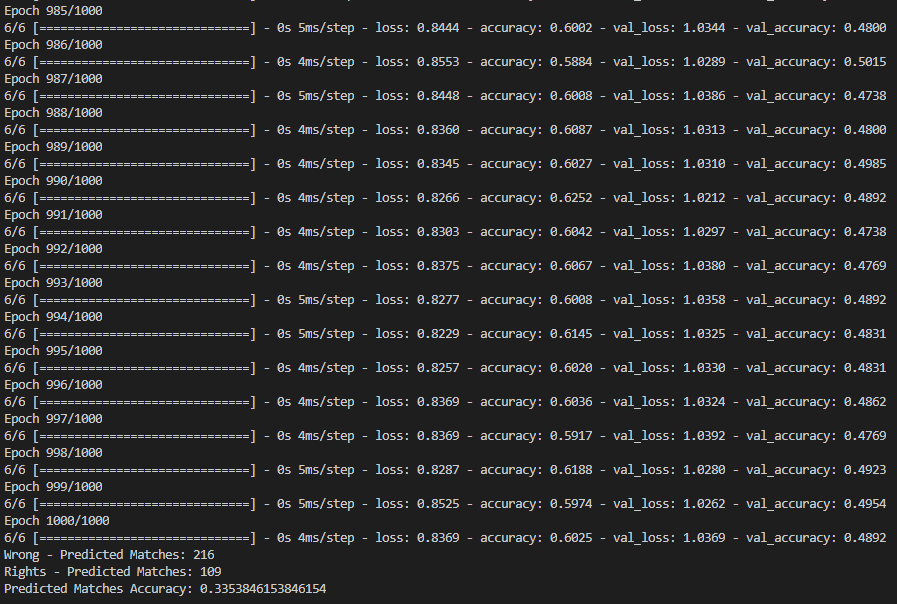
Εικόνα Εφαρμογής 19



Εικόνα Εφαρμογής 20



Εικόνα Εφαρμογής 21



Εικόνα Εφαρμογής 22