ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ακαδημαϊκό έτος: **2020-2021**
Εξεταστική: **Σεπτέμβριος**
Μάθημα: **Αναγνώριση Προτύπων**
Εξάμηνο: **5ο**


| **Ονοματεπώνυμο** | **ΑΜ** |
|---|---|
| Μπαντάνα Δανάη Ιωάννα | Π17081 |
| Ρούντου Άννα Φανή | Π17113 |
| Σαγιέντ Ιωσήφ | Π15123 |


**«Αρχείο Συνοπτικής Παρουσίασης των Κύριων Σημείων της Εργασίας»**

# Περιεχόμενα

# Εισαγωγή

Για την υλοποίηση κοινών κοινών μεθόδων και συναρτήσεων δημιουργήθηκαν δύο αρχεία python, το «Data» και το «Functions». Το πρώτο αρχείο «Data» είναι υπεύθυνο για την φόρτωση των δεδομένων από τη βάση δεδομένων και για τον μετέπειτα χειρισμό τους στην κατάλληλη μορφή. Το δεύτερο αρχείο «Functions» περιέχει τις ωφέλιμες και καθολικής χρήσης συναρτήσεις και για τα τρία ερωτήματα. Με τον τρόπο αυτό επιτυγχάνεται η καλύτερη ανάγνωση και δόμηση του κώδικα.

**Γενικές έννοιες**

*1. Διαχείριση Δεδομένων*

Η διαχείριση των δεδομένων γίνεται στο αρχείο «Data.py». Απαιτείται κατάλληλος χειρισμός για την δημιουργία πινάκων με τα συγκεκριμένα δεδομένα που ζητούνται για κάθε ερώτημα. Αρχικά ορίζονται όλοι οι πίνακες που θα χρησιμοποιηθούν.

Για τα ερωτήματα i και ii καλείται η μέθοδος «loadData» στη οποία γίνεται ανάγνωση του αρχείου «Match.csv» προκειμένου να αποθηκευτούν στον πίνακα «Match» τα δεδομένα των αγώνων. Όπως τονίστηκε στην εκφώνηση, έχει ήδη πραγματοποιηθεί μία επεξεργασία στο σύνολο των δεδομένων «Match» με την οποία αφαιρέθηκαν οι εγγραφές που τα αντίστοιχα διανύσματα προγνωστικών έχουν μηδενικές τιμές. Έτσι, το «Match.csv» το οποίο διαβάζει το αρχείο μας δεν περιέχει τα πεδία αυτά και έχει μέγεθος 22592 εγγραφές. Στη συνέχεια, για κάθε αγώνα υπολογίζεται ποιος νίκησε με τη βοήθεια μίας συνάρτησης του αρχείου «Functions.py» την «match_result(home_goals,away_goals)». Το αποτέλεσμα μπορεί να είναι «H», «D» ή «A» τα οποία αντιστοιχούν στο αποτέλεσμα αγώνα «Νίκη Εντός», «Ισοπαλία», «Νίκη Εκτός». Το αποτέλεσμα αυτό μαζί με το id του αγώνα συντάσσουν τον πίνακα «Match_Results». Ύστερα, δημιουργείται ο δυσδιάστατος πίνακας «Companies_odds» που περιέχει για κάθε στοιχηματική εταιρία τα δεδομένα των αποδόσεων τους για κάθε αγώνα.

Για το ερώτημα iii καλείται η μέθοδος «loadMLPData» όπου εκτός από τον πίνακα των αγώνων «Match» χρειαζόμαστε και τα χαρακτηριστικά των ομάδων του κάθε αγώνα, που τα παίρνουμε από το αρχείο «TeamAttributes.csv». Για κάθε ομάδα υπολογίζουμε από τη χρονολογία της τη σεζόν στην οποία συμμετείχε και τοποθετούμε στον πίνακα «TeamAttributes», τη σεζόν, το id της ομάδας και τα υπόλοιπα χαρακτηριστικά που μας ζητήθηκαν, δηλαδή τα buildUpPlaySpeed, buildUpPlayPassing, chanceCreationPassing, chanceCreationCrossing, chanceCreationShooting, defencePressure, defenceAggregation και το defenceTeamWidth. Έπειτα, για κάθε αγώνα του πίνακα «Match» καθορίζονται τα id των ομάδων που αγωνίζονται καθώς και τη σεζόν του αγώνα. Σύμφωνα με τα παραπάνω, αναζητούνται τα χαρακτηριστικά των συμμετεχουσών ομάδων στον πίνακα «TeamAttributes». Εάν βρεθούν αντίστοιχες εγγραφές τότε αυτές

αποθηκεύονται στους πίνακες «HomeAttributes» και «AwayAttributes» που περιγράφουν τα χαρακτηριστικά των ομάδων «Εντός» και «Εκτός» αντίστοιχα. Άμα και για τις δύο ομάδες βρεθούν χαρακτηριστικά, τότε δημιουργούμε το πλήρες διάνυσμα χαρακτηριστικών του κάθε αγώνα που περιέχει τα χαρακτηριστικά των δύο ομάδων καθώς και τις αποδόσεις των στοιχηματικών εταιριών για τον εκάστοτε αγώνα. Το πλήρες αυτό διάνυσμα αποθηκεύεται στον πίνακα «MatchFeatures». Τέλος, ανάλογα με το αποτέλεσμα του αγώνα, που υπολογίζεται μέσω της συνάρτησης «match_result(home_goals,away_goals)», και με τη βοήθεια μίας άλλης συνάρτησης της «cal_output(result)», που αντιστοιχεί το αποτέλεσμα «H», «D» και «A» του αγώνα σε ένα διάνυσμα μεγέθους n = 3 το οποίο περιέχει την τιμή 1 στην πραγματική έκβαση του αγώνα και την τιμή 0 στις υπόλοιπες. Δηλαδή, για το αποτέλεσμα «Νίκη Εκτός» η μεταβλητή «result» θα πάρει τιμή «A» και επομένως θα επιστραφεί το διάνυσμα [0,0,1]. Τα διανύσματα αυτά θα αποτελέσουν τα δεδομένα του πίνακα «MatchFeaturesOutput». Η παραπάνω επεξεργασία μας δίνει ένα σύνολο 15.484 αγώνων, όπου ο κάθε ένας έχει 28 παραμέτρους, και ένα αντίστοιχου μεγέθους σύνολο των αποτελεσμάτων των αγώνων.

2. *Μέθοδος k-πλης Διεπικύρωσης (k-fold-cross-validation)*

Η μέθοδος αυτή αναφέρεται στον «k» φορές επαναληπτικό χωρισμό του συνόλου δεδομένων σε σύνολα για εκπαίδευση(training) και για δοκιμή(testing). Σε κάθε επανάληψη(fold) τα σύνολα αυτά είναι διαφορετικά. Για την εργασία αποφασίσαμε ότι θα κάνουμε 10 επαναλήψεις, δηλαδή θα εφαρμόσουμε τη μέθοδο της 10-πλης διεπικύρωσης, με το 90% των δεδομένων να χρησιμοποιείται για εκπαίδευση και το 10% των δεδομένων για δοκιμή σε κάθε επανάληψη. Φυσικό είναι κάποια δεδομένα να χαθούν, αφού χωρίζουμε το σύνολο δεδομένων σε 10 ισόποσες ομάδες. Επομένως, το σύνολο των δεδομένων για δοκιμή θα είναι κάθε φορά 2259 για το ερώτημα i και ii και 1548 για το ερώτημα iii. Σε κάθε επανάληψη επιλέγεται για το σύνολο δοκιμής το σειριακό επόμενο σύνολο των 2259 και 1548 δεδομένων. Αντίστοιχα, σε κάθε επανάληψη στο σύνολο εκπαίδευσης τοποθετούνται τα υπόλοιπα δεδομένα. Η διαδικασία αυτή πραγματοποιείται στη συνάρτηση «k_fold_cross_validation(inputs,k)» του αρχείου «Functions.py». Η συνάρτηση αυτή δέχεται το σύνολο όλων των δεδομένων μας και τον αριθμό των επαναλήψεων που θέλουμε να κάνουμε (k). Μας επιστρέφει δύο πίνακες, το «training_set» και τον «testing_set», καθένας διάστασης 10 θέσεων, που περιέχουν κάθε πιθανό συνδυασμό δεδομένων εκπαίδευσης και δοκιμής για και τις 10 επαναλήψεις.

3. *Συναρτήσεις*

Στο αρχείο «Functions.py» συναντάμε όλες τις συναρτήσεις που θα χρησιμοποιηθούν από τα ερωτήματα.

- ➔ 'match_result(home_goals, away_goals)': Υπολογίζει, σύμφωνα με τον αριθμό των γκολ της εντός ομάδας και εκτός ποια νίκησε και επιστρέφει «H» εάν νίκησε η εντός ομάδα, «D» για ισοπαλία και «A» για τη νίκη της εκτός ομάδας.
- ➔ 'k_fold_cross_validation(inputs,k)': Υλοποιεί τη μέθοδο της 10-πλής διεπικύρωσης (10 fold cross validation). Χωρίζει το σύνολο των δεδομένων σε k κομμάτια(folds) και κατασκευάζει για κάθε fold, ένα σετ δεδομένων για εκπαίδευση και ένα σετ δεδομένων για δοκιμή. Αυτά τα σετ αποθηκεύονται στο «training_set» και «testing_set» αντίστοιχα. Τελικά μας επιστρέφει αυτούς τους δύο πίνακες που θα περιέχουν 10 training_sets και αντίστοιχα 10 testing sets. Η διαδικασία «k-fold-cross-validation» έχει αναλυθεί αναλυτικά παραπάνω.
- ➔ 'modify_X(company_odds)': Μετασχηματίζει την είσοδο X, που περιέχει κάθε φορά τα στατιστικά μίας εταιρίας, εισάγοντας στη πρώτη θέση την τιμή 1. Η τιμή αυτή αντιπροσωπεύει το 'x0' στο διάνυσμα X των χαρακτηριστικών.
- ➔ 'desired_values(fold,each_fold,results,class_i,total_matches,not_class_i_symbol)': Κατασκευάζει και επιστρέφει ένα πίνακα με τις επιθυμητές εξόδους του ταξινομητή για ένα fold και τη δηλωμένη κλάση(class_i=H,D ή A). Αγνοεί να υπολογίσει τις επιθυμητές τιμές για το σύνολο των δεδομένων που ανήκουν στο σετ της δοκιμής (testing) του fold. Τοποθετεί την τιμή 1 αν ισχύει η έκβαση η οποία ψάχνουμε και την τιμή 0 για το ερώτημα i και την τιμή -1 για το ερώτημα ii για τις άλλες περιπτώσεις. Δηλαδή, αν ψάχνουμε το υπερεπίπεδο το οποίο χωρίζει την κλάση «Νίκη Εντός» («H») από τις άλλες δύο, τότε ο πίνακας «d» θα έχει την τιμή 1 σε κάθε match του οποίου η έκβαση αποτελέσματος ήταν «Νίκη Εντός». Οι αντίστοιχες θέσεις των υπόλοιπων αγώνων θα έχουν τιμή -1.
- ➔ 'y_output(X,match,W)': Υπολογίζει την πραγματική έξοδο που δίνει ο ταξινομητής.
- ➔ 'lms_robbins_monro(X,d)': Υλοποιεί τον αλγόριθμο «Ελάχιστου Μέσου Τετραγωνικού Σφάλματος» με την αναπροσαρμογή των βαρών σύμφωνα με τη μεθοδολογία «Robbins-Monro». Δέχεται το σύνολο εκπαίδευσης μίας επανάληψης/fold και τις αντίστοιχες πραγματικές εκβάσεις αυτών και επιστρέφει το βέλτιστο διάνυσμα βαρών για την συγκεκριμένη εξίσωση. Πιο αναλυτικά, υπολογίζοντας την συνάρτηση κόστους και μαζί με τον σταθερό όρο γίνεται η προσαρμογή των βαρών για το σύνολο δεδομένων «X». Η προσαρμογή θα τελειώσει είτε όταν τελειώσουν τα δεδομένα είτε όταν η ο σταθερός όρος 'α' πάρει τιμή μικρότερη ή ίση του 0.0000001. Το τελευταίο δηλώνει ότι η σύγκλιση του ταξινομητή επιτεύχθηκε, προσεγγίζοντας τις επιθυμητές τιμές.

- ➔ ‘score_classifier(data,W,fold,results)’: Στο σύνολο δοκιμής μίας επανάληψης/fold (data), εφαρμόζονται τα αντίστοιχα βάρη (W) που βρέθηκαν με την «lms_robbins_monro» και υπολογίζεται μία πρόβλεψη για κάθε ένα δεδομένο του συνόλου δοκιμής. Η πρόβλεψη αυτή υπολογίζεται εκτελώντας την συνάρτηση πρόβλεψης και υπολογίζοντας από αυτή το κόστος της. Κατά συνέπεια στη θέση του ελάχιστου κόστους θα βρίσκεται η καλύτερη πρόβλεψη για κάθε δεδομένο του συγκεκριμένου fold και συνόλου δοκιμής. Δέχεται και τον πίνακα με τα πραγματικά αποτελέσματα των αγώνων και τα συγκρίνει με τις τιμές πρόβλεψης που προέκυψαν. Αν η πρόβλεψη για ένα δεδομένο του συνόλου δοκιμής είναι σωστή, τότε ένας μετρητής «correct» αυξάνεται κατά 1. Τέλος, επιστρέφεται ο αριθμός των σωστών προβλέψεων της ταξινόμησης.
- ➔ ‘Cal_output(output)’: Επιστέφει το διάνυσμα που αντιστοιχεί στην είσοδο «output».

# 1. Ερώτημα i

Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Μέσου Τετραγωνικού Σφάλματος (**Least Mean Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφής $\boldsymbol{g_k(\psi_\kappa(m))}$: $\mathbb{R}^3 \rightarrow \{\boldsymbol{H}, \boldsymbol{D}, \boldsymbol{A}\}$ για κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.

## 1.1. Υλοποίηση

Για την υλοποίηση του «LMS» (**Least Mean Squares**) χρησιμοποιούμε συναρτήσεις που περιέχονται στο python αρχείο «Functions.py». Παραπάνω, έχουν αναλυθεί λεπτομερώς οι συναρτήσεις που θα χρησιμοποιήσουμε. Η διαδικασία της εκπαίδευσης και της δοκιμής του ταξινομητή μας περιέχεται στο αρχείο «LeastMeanSquares.py».

Αρχικά, φορτώνουμε τα δεδομένα που θα χρειαστούμε σε πίνακες με τη βοήθεια της μεθόδου «loadData» του αρχείου «Data.py».

***Εκπαίδευση του Ταξινομητή***

Στη συνέχεια, θέλουμε να υπολογίσουμε για κάθε στοιχηματική εταιρία και για κάθε fold της μεθόδου της 10-πλής διεπικύρωσης (10 fold cross validation) ένα διάνυσμα βαρών. Αυτό το επιτυγχάνουμε καλώντας την «k_fold_cross_validation» για κάθε στοιχηματική. Έτσι παίρνουμε:

- Για κάθε στοιχηματική 10 σύνολα δεδομένων εκπαίδευσης και τα αντίστοιχα 10 σύνολα δοκιμής.
    - Για κάθε επανάληψη/fold τροποποιούμε το σύνολο δεδομένων εκπαίδευσης με τη συνάρτηση «modify_X». Τώρα έχουμε τα δεδομένα εκπαίδευσης έτοιμα για την διαδικασία εκπαίδευσης του ταξινομητή.
        - Για κάθε μία από τις πιθανές εκβάσεις «H», «D» και «A» με τη βοήθεια της συνάρτησης «desired_values» υπολογίζονται τα πραγματικά αποτελέσματα των αγώνων «d». Στην συνέχεια, υπολογίζονται με τη συνάρτηση «lms_robbins_monro(X,d)» τα βέλτιστα βάρη για τη συγκεκριμένη έκβαση κάθε fold.

      Έτσι, έχουμε υπολογίσει τρία σετ βαρών για κάθε fold.

Η λεπτομερής λειτουργία των παραπάνω συναρτήσεων έχει αναλυθεί προηγουμένως.

Με το πέρας των παραπάνω συναρτήσεων έχουμε στην διάθεσή μας τα σύνολα εκπαίδευσης «training_sets», τα σύνολα δοκιμής «testing_sets» και τα βάρη «company_fold_weight» για κάθε fold για κάθε στοιχηματική εταιρία.

***Δοκιμή του Ταξινομητή***

Θέλουμε να υπολογίσουμε το πόσες σωστές ταξινομήσεις θα κάνει ο ταξινομητής για κάθε στοιχηματική εταιρεία για κάθε fold.

- Για κάθε στοιχηματική εταιρία
    - Για κάθε fold υπολογίζεται με τη βοήθεια της συνάρτησης «score_classifier» ο αριθμός των σωστών προβλέψεων που υπολόγισε ο ταξινομητής και τον αποθηκεύουμε στον πίνακα «scores».

Τέλος, για να εκτιμήσουμε την στοιχηματική εταιρεία, τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης, πρώτα πρώτα θα εκτιμήσουμε το καλύτερο σετ βαρών και την αποδοτικότητα του ταξινομητή μας για κάθε εταιρεία.

***Εκτίμηση στοιχηματικής εταιρίας με μεγαλύτερη ακρίβεια ταξινόμησης***

Θα βρούμε το καλύτερο σετ βαρών μεταξύ όλων των folds για τη κάθε εταιρία.

- Για κάθε στοιχηματική εταιρεία
    - Βρίσκουμε τον μέγιστο αριθμό των σωστών προβλέψεων μεταξύ των προβλέψεων όλων των folds, άρα βρίσκουμε το fold με την καλύτερη επίδοση.

Έτσι, κρατάμε για κάθε εταιρία το fold με την καλύτερη πρόβλεψη, την πρόβλεψη αυτή καθώς και τα βάρη που χρησιμοποιήθηκαν στην συνάρτηση πρόβλεψης και επέφεραν την μέγιστη απόδοση του ταξινομητή.

Τέλος, συγκρίνουμε τα καλύτερα σετ βαρών των εταιριών μεταξύ τους και βρίσκουμε το σετ βαρών και την εταιρία που οδήγησε στη μεγαλύτερη ακρίβεια ταξινόμησης.

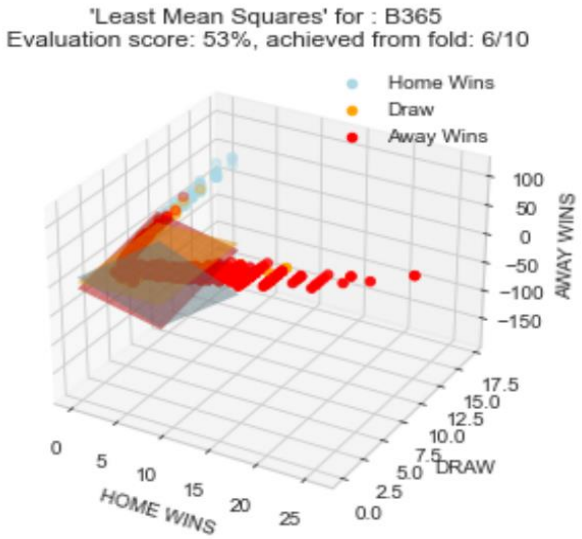
## 1.2. Εκτέλεση

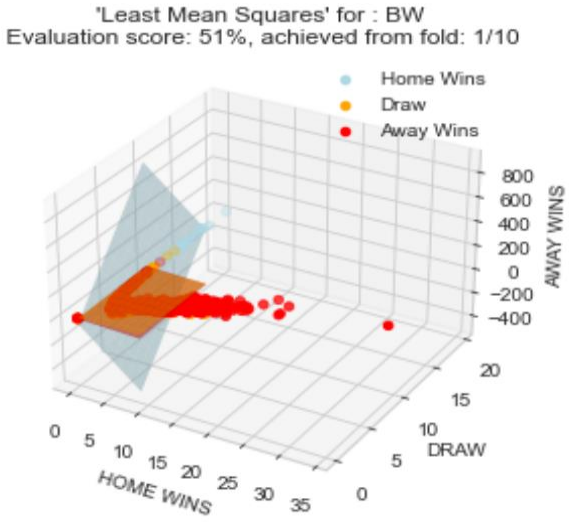
Η γραφική αναπαράσταση των παραπάνω υπερεπιπέδων πραγματοποιήθηκε με τη βιβλιοθήκη matplotlib. Η εκτέλεση και τα αποτελέσματα του «LeastMeanSquares.py»:

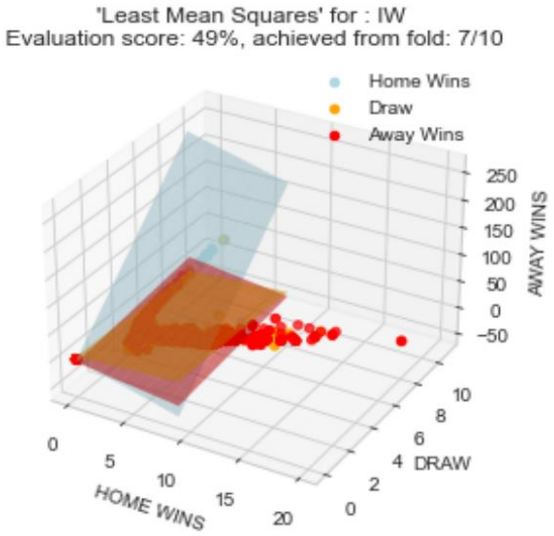
***Η συγκεντρωτική αποτύπωση της ταξινομητικής ακρίβειας των διάφορων ταξινομητών ανά στοιχηματική εταιρεία και cross validation fold.***

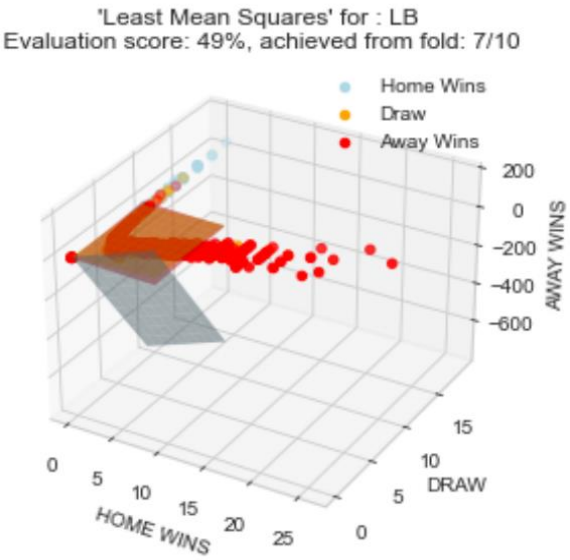
```
Company: B365 Accuracy: 53 %
Company: BW Accuracy: 51 %
Company: IW Accuracy: 49 %
Company: LB Accuracy: 49 %

Best betting company: B365
Best fold: 6
Accuracy: 53 %

Best weights for company:  B365
For  HOME :  [-0.10143529191476489, -0.09891471411721486, 0.19224287732339823, 0.008946184123527455]
For  DRAW :  [-2.039768165399374, -0.07713067158045198, -0.24294416564579419, 0.7146178336421545]
For  AWAY :  [0.34461792622930587, 0.15391929067783988, -0.16968746843292637, 0.045718400906923404]


Best weights for company:  BW
For  HOME :  [-0.09159601369060147, -0.09120649773914627, 0.20124200567051648, -0.0019590592589677396]
For  DRAW :  [-2.268963474711679, -0.1810333911740356, -0.16761764002914314, 0.7371142391553126]
For  AWAY :  [0.3160186186850902, 0.1305782343073037, -0.1681267196621347, 0.06077314446522976]


Best weights for company:  IW
For  HOME :  [-0.007682682808626862, -0.026703813374038265, 0.14729067240900146, -0.005092660075048884]
For  DRAW :  [-3.0750651927847232, -0.8235959101729419, 0.6407822782459842, 0.6486636515252018]
For  AWAY :  [0.33935274700034923, 0.14634147257766186, -0.17934040157761025, 0.05757160970644543]


Best weights for company:  LB
For  HOME :  [-0.007109731182007662, -0.026341302128960332, 0.13239040049467046, 0.0015708452082428062]
For  DRAW :  [-2.9419347017081954, -0.7101692892025848, 0.5428521921834666, 0.5757560403843637]
For  AWAY :  [0.277042138260067, 0.09863255757042984, -0.13771377394058454, 0.061942948009688345]
```

***Οπτικοποίηση των εμπλεκόμενων προβλημάτων ταξινόμησης σε διαγράμματα.***



# 2. Ερώτημα ii

Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Τετραγωνικού Σφάλματος (**Least Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφής $\boldsymbol{g_k(\psi_\kappa(m))}$: $\mathbb{R}^3 \rightarrow \{\boldsymbol{H}, \boldsymbol{D}, \boldsymbol{A}\}$ για κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.

## 2.1. Υλοποίηση

Για την υλοποίηση του «LS» (**Least Squares**) ακολουθείται ακριβώς η ίδια διαδικασία με το την υλοποίηση του «LMS». Επομένως, χρησιμοποιούμε συναρτήσεις που περιέχονται στο python αρχείο «Functions.py» και οι οποίες έχουν αναλυθεί λεπτομερώς παραπάνω. Η διαδικασία της εκπαίδευσης και της δοκιμής του ταξινομητή μας περιέχεται στο αρχείο «LeastSquares.py».

Αρχικά, φορτώνουμε τα δεδομένα που θα χρειαστούμε σε πίνακες με τη βοήθεια της μεθόδου «loadData» του αρχείου «Data.py».

***Εκπαίδευση του Ταξινομητή***

Στη συνέχεια, θέλουμε να υπολογίσουμε για κάθε στοιχηματική εταιρία και για κάθε fold της μεθόδου της 10-πλής διεπικύρωσης (10 fold cross validation) ένα διάνυσμα βαρών. Αυτό το επιτυγχάνουμε καλώντας την «k_fold_cross_validation» για κάθε στοιχηματική. Έτσι παίρνουμε:

- Για κάθε στοιχηματική 10 σύνολα δεδομένων εκπαίδευσης και τα αντίστοιχα 10 σύνολα δοκιμής.
    - Για κάθε επανάληψη/fold τροποποιούμε το σύνολο δεδομένων εκπαίδευσης με τη συνάρτηση «modify_X». Τώρα έχουμε τα δεδομένα εκπαίδευσης έτοιμα για την διαδικασία εκπαίδευσης του ταξινομητή.
        - Για κάθε μία από τις πιθανές εκβάσεις «H», «D» και «A» με τη βοήθεια της συνάρτησης «desired_values» υπολογίζονται τα πραγματικά αποτελέσματα των αγώνων «d». Στην συνέχεια, τα βέλτιστα βάρη για τη συγκεκριμένη έκβαση κάθε fold υπολογίζονται λύνοντας το σύστημα εξισώσεων «w_i = np.linalg.inv(X.T @ X) @ (X.T @ d)». Ο διαφορετικός

τρόπος σύγκλισης στα βέλτιστα βάρη αποτελεί το **σημείο διαφοράς** μεταξύ του «LS» και του «LMS» στην εργασία μας.

Έτσι, έχουμε υπολογίσει τρία σετ βαρών για κάθε fold.

Η λεπτομερής λειτουργία των παραπάνω συναρτήσεων έχει αναλυθεί προηγουμένως.

Με το πέρας των παραπάνω συναρτήσεων έχουμε στην διάθεσή μας τα σύνολα εκπαίδευσης «training_sets», τα σύνολα δοκιμής «testing_sets» και τα βάρη «company_fold_weight» για κάθε fold για κάθε στοιχηματική εταιρία.

***Δοκιμή του Ταξινομητή***

Θέλουμε να υπολογίσουμε το πόσες σωστές ταξινομήσεις θα κάνει ο ταξινομητής για κάθε στοιχηματική εταιρεία για κάθε fold.

- Για κάθε στοιχηματική εταιρία
    - Για κάθε fold υπολογίζεται με τη βοήθεια της συνάρτησης «score_classifier» ο αριθμός των σωστών προβλέψεων που υπολόγισε ο ταξινομητής και τον αποθηκεύουμε στον πίνακα «scores».

Τέλος, για να εκτιμήσουμε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης, πρώτα πρώτα θα εκτιμήσουμε το καλύτερο σετ βαρών και την αποδοτικότητα του ταξινομητή μας για κάθε εταιρεία.

***Εκτίμηση στοιχηματικής εταιρίας με μεγαλύτερη ακρίβεια ταξινόμησης***

Θα βρούμε το καλύτερο σετ βαρών μεταξύ όλων των folds για τη κάθε εταιρία.

- Για κάθε στοιχηματική εταιρεία
    - Βρίσκουμε τον μέγιστο αριθμό των σωστών προβλέψεων μεταξύ των προβλέψεων όλων των folds, άρα βρίσκουμε το fold με την καλύτερη επίδοση.

Έτσι, κρατάμε για κάθε εταιρία το fold με την καλύτερη πρόβλεψη, την πρόβλεψη αυτή καθώς και τα βάρη που χρησιμοποιήθηκαν στην συνάρτηση πρόβλεψης και επέφεραν την μέγιστη απόδοση του ταξινομητή.

Τέλος, συγκρίνουμε τα καλύτερα σετ βαρών των εταιριών μεταξύ τους και βρίσκουμε το σετ βαρών και την εταιρία που οδήγησε στη μεγαλύτερη ακρίβεια ταξινόμησης.

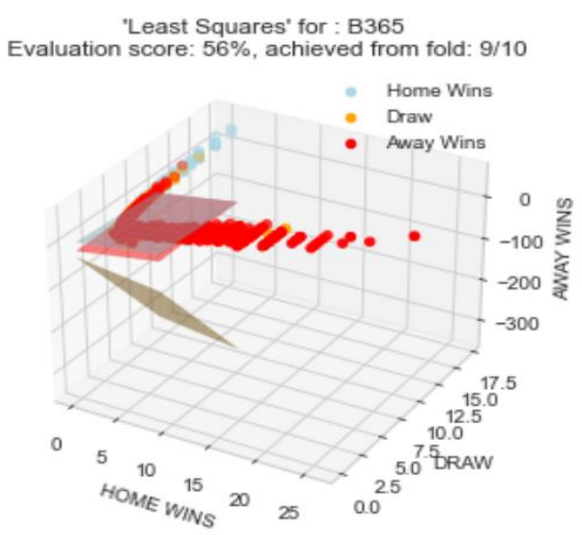
## 2.2. Εκτέλεση

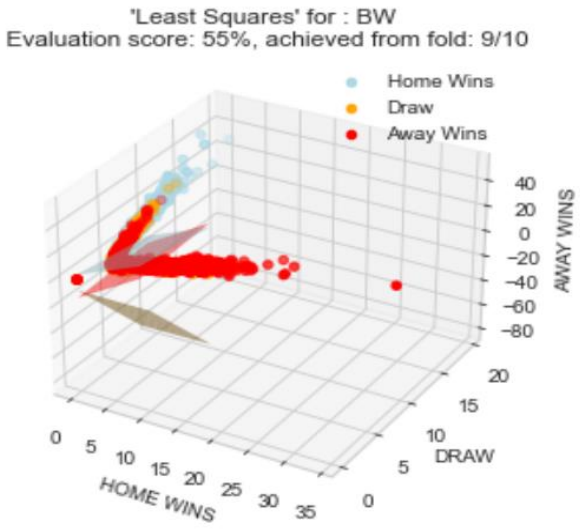
Η γραφική αναπαράσταση των παραπάνω υπερεπιπέδων πραγματοποιήθηκε με τη βιβλιοθήκη matplotlib. Η εκτέλεση και τα αποτελέσματα του «LeastSquares.py»:

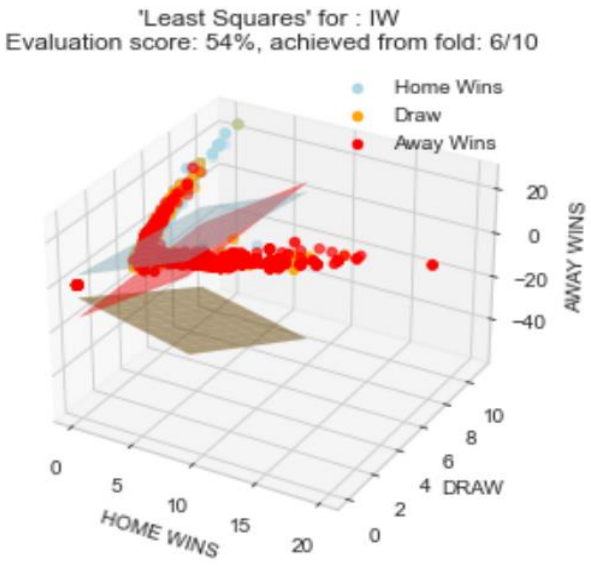
***Η συγκεντρωτική αποτύπωση της ταξινομητικής ακρίβειας των διάφορων ταξινομητών ανά στοιχηματική εταιρεία και cross validation fold.***

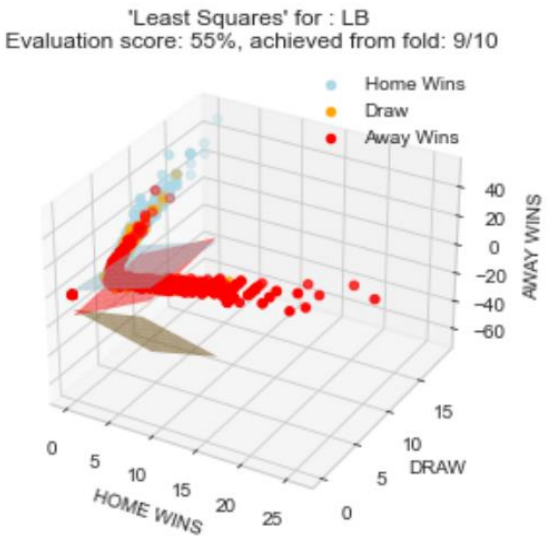
```
Company: B365 Accuracy: 56 %
Company: BW Accuracy: 55 %
Company: IW Accuracy: 54 %
Company: LB Accuracy: 55 %

Best betting company:  B365
Best fold:  9
Best percentage:  56 %

Best weights for company:  B365
For  HOME :  [-0.20622765 -0.12028892  0.06168162  0.04713959]
For  DRAW :  [-0.092429   -0.00946587 -0.09477072 -0.00303637]
For  AWAY :  [-0.70134335  0.12975479  0.0330891  -0.04410322]


Best weights for company:  BW
For  HOME :  [-0.23080628 -0.12872365  0.05861634  0.05873433]
For  DRAW :  [-0.10296793 -0.01853851 -0.0776956  -0.01211821]
For  AWAY :  [-0.6662258   0.14726216  0.01907926 -0.04661612]


Best weights for company:  IW
For  HOME :  [-0.26163094 -0.13419267  0.0713379   0.06093243]
For  DRAW :  [-0.09966078 -0.02147828 -0.07441127 -0.01741428]
For  AWAY :  [-0.63870828  0.15567095  0.00307337 -0.04351815]


Best weights for company:  LB
For  HOME :  [-0.16813725 -0.12166503  0.03251159  0.06214011]
For  DRAW :  [-0.13976229 -0.0195014  -0.06602414 -0.01384454]
For  AWAY :  [-0.69210047  0.14116643  0.03351254 -0.04829557]
```

***Οπτικοποίηση των εμπλεκόμενων προβλημάτων ταξινόμησης σε διαγράμματα.***



# 3. Ερώτημα iii

Να υλοποιήσετε ένα πολυστρωματικό νευρωνικό δίκτυο, ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί μια συνάρτηση διάκρισης της μορφής $\boldsymbol{g}(\boldsymbol{\Phi}(\boldsymbol{m}))$: $\mathbb{R}^{\mathbf{28}} \rightarrow \{\boldsymbol{H}, \boldsymbol{D}, \boldsymbol{A}\}$, όπου το $\boldsymbol{\Phi}(\boldsymbol{m}) \in \mathbb{R}^{\mathbf{28}}$ αντιστοιχεί στο πλήρες διάνυσμα χαρακτηριστικών του κάθε αγώνα που δίνεται από την σχέση:

$$\boldsymbol{\Phi}(\boldsymbol{m}) = [\boldsymbol{\varphi}(\boldsymbol{h}), \boldsymbol{\varphi}(\boldsymbol{\alpha}), \boldsymbol{\psi}_{\boldsymbol{B365}}(\boldsymbol{m}), \boldsymbol{\psi}_{\boldsymbol{BW}}(\boldsymbol{m}), \boldsymbol{\psi}_{\boldsymbol{IW}}(\boldsymbol{m}), \boldsymbol{\psi}_{\boldsymbol{LW}}(\boldsymbol{m})]$$

## 3.1. Υλοποίηση

Για την υλοποίηση του πολυστρωματικού νευρωνικού δικτύου φτιάξαμε ένα python αρχείο «MLP.py» στο οποίο χρησιμοποιήθηκαν οι βιβλιοθήκες «sklearn.neural_network», «sklearn.metrics» και «sklearn.preprocessing». Η κλάση «MLPClassifier» της «sklearn.neural_network» βιβλιοθήκης υλοποιεί έναν MLP πολλαπλών στρωμάτων ο οποίος εκπαιδεύεται χρησιμοποιώντας «Backpropagation».

Αρχικά, φορτώνουμε τα δεδομένα που χρειαζόμαστε στους πίνακες «MatchFeatures» και «MatchOutputs» μέσω της συνάρτησης «loadMLPData» του αρχείου «Data.py». Ο πρώτος περιέχει τα πλήρη διανύσματα χαρακτηριστικών του κάθε αγώνα, ενώ ο δεύτερος αποτυπώνει το πραγματικό αποτέλεσμα κάθε αγώνα σε μορφή διανύσματος μεγέθους ν = 3. Στη συνέχεια, καλείται η «k_fold_cross_validation» του αρχείου «Functions.py» για και τους δύο προαναφερθέντες πίνακες. Με τον τρόπο αυτό επιτυγχάνουμε να φτιάξουμε 10 σύνολα εκπαίδευσης «training_sets» με τα αντίστοιχα 10 σύνολα δοκιμής «testing_sets». Μαζί με αυτά, φτιάχνουμε τα αντίστοιχα σετ των πραγματικών αποτελεσμάτων των δεδομένων εκπαίδευσης «training_outputs» και των δεδομένων δοκιμής «testing_outputs». Τα σύνολα των δεδομένων και των αποτελεσμάτων ταυτίζονται πλήρως. Έπειτα, δημιουργούμε ένα αντικείμενο «MLPClassifier» με τις εξής παραμέτρους:

```
MLP = MLPClassifier(solver='sgd', activation='logistic', batch_size=54, learning_rate_init=
0.01, momentum=0.9, hidden_layer_sizes=(12, 3), random_state=1, max_iter=200)
```

Αναλυτικά οι παράμετροι και τα χαρακτηριστικά:

- ➔ 'Solver="sgd"': αναφέρεται στην στοχαστική κλίση (stochastic gradient descent)
- ➔ 'Activation= "logistic"': αναφέρεται στην σιγμοειδή συνάρτηση logistic, η οποία επιστρέφει f(x) = 1 / (1 + exp(-x))

➔ ‘batch_size’: Το μέγεθος των μίνι παρτίδων για τους στοχαστικούς βελτιστοποιητές.

➔ ‘learning_rate_init’: Ελέγχει το μέγεθος του βήματος κατά ενημέρωση των βαρών.

➔ ‘momentum’: Ενημερώνει την κλίση (gradient descent). Η κλίση αυτή πρέπει να είναι μεταξύ 0 και 1.

➔ ‘hidden_layer_sizes’: Δηλώνει τον αριθμό των κρυφών στρωμμάτων καθώς και τον αριθμό των νευρώνων κάθε κρυφού στρώματος. Δηλαδή, το n-οστό στοιχείο αντιπροσωπεύει τον αριθμό των νευρώνων στο n-οστό κρυφό στρώμα.

➔ ‘random_state’: Καθορίζει τη δημιουργία των τυχαίων αριθμών για τα βάρη και το bias.

➔ ‘max_iter’: Μέγιστος αριθμός επαναλήψεων. Ο solver επαναλαμβάνεται μέχρι αυτόν τον αριθμό επαναλήψεων. Για στοχαστικούς solvers («sgd», «adam»), αυτό καθορίζει τον αριθμό των epochs (πόσες φορές θα χρησιμοποιηθεί κάθε data point) και όχι τον αριθμό των βημάτων κλίσης (gradient steps).

Από το πολυστρωματικό νευρωνικό δίκτυο «MLP» που δημιουργήσαμε, θα χρησιμοποιήσουμε 2 από τις μεθόδους του, την «fit» και την «predict». Η πρώτη προσαρμόζει το μοντέλο στα δεδομένα εκπαίδευσης και στις τιμές στόχους. Η δεύτερη προβλέπει τιμές βάσει ενός συνόλου δοκιμής, χρησιμοποιώντας τον ταξινομητή «πολλαπλών επιπέδων perceptron/MLP».

Το παραπάνω αντικείμενο «MLP» αποτελεί τον ταξινομητή μας για το ερώτημα iii.

Εφαρμόζουμε τη μέθοδο της 10-πλής διεπικύρωσης (10 fold cross validation).

- Για κάθε διεπικύρωση/fold
    - Κανονικοποιούμε τα δεδομένα προς εκπαίδευση (x_train) και μαζί με το αντίστοιχο σετ αποτελεσμάτων (y_train), τα περνάμε ως ορίσματα στη συνάρτηση «fit» του «MLPClassifier» ώστε να γίνει η εκπαίδευση του ταξινομητή.

! Η κανονικοποίηση πραγματοποιείται διότι το MLP είναι ευαίσθητο ως προς τα δεδομένα του. Στη συνέχεια,

- Κανονικοποιούμε τα δεδομένα δοκιμής «x_test», ορίζουμε το αντίστοιχο σετ αποτελεσμάτων «y_test» και εκτελώντας τη μέθοδο «predict» του «MLPClassifier» με όρισμα τα δεδομένα δοκιμής «x_test» παίρνουμε τις τιμές πρόβλεψης του ταξινομητή «y_pred».
- Με τη συνάρτηση «accuracy_score» της βιβλιοθήκης «sklearn.metrics» γίνεται ο υπολογισμός της ακρίβειας/απόδοσης του ταξινομητή για την

εκάστοτε επανάληψη/fold. Τα ορίσματα της μεθόδου αποτελούν τα πραγματικά αποτελέσματα «y_test» και οι τιμές πρόβλεψης «y_pred».

Κατά το πέρας κάθε επανάληψης/fold εμφανίζεται η ακρίβεια της και στο τέλος υπολογίζουμε τη μέγιστη ακρίβεια και από ποιό fold προήλθε.

## 3.2. Εκτέλεση

Το παρακάτω παράδειγμα εκτέλεσης λειτουργεί με τις εξής παραμέτρους για το «MLP»:

- inputs = 28
- solver='sgd'
- activation='logistic'
- batch_size=54
- learning_rate_init=0.01
- momentum=0.9
- hidden_layer_sizes=(10, 3)
    - hidden_layer1 = 10
    - hidden_layer2 = 3
- random_state=1
- max_iter=300

***Η συγκεντρωτική αποτύπωση της ταξινομητικής ακρίβειας του MLPClassifier ανά cross-fold-validation.***

```
Fold: 1 Accuracy: 0.3029715762273902
Fold: 2 Accuracy: 0.32493540051679587
Fold: 3 Accuracy: 0.24741602067183463
Fold: 4 Accuracy: 0.229328165374677
Fold: 5 Accuracy: 0.2771317829457364
Fold: 6 Accuracy: 0.28811369509043927
Fold: 7 Accuracy: 0.3578811369509044
Fold: 8 Accuracy: 0.34366925064599485
Fold: 9 Accuracy: 0.28036175710594313
Fold: 10 Accuracy: 0.35658914728682173

The most accurate prediction came from fold  7  with prediction accuracy:
35.78811369509044 %
```