Θεωρία Αποφάσεων

Νάνος Νικόλαος 1054386
Ζερβός Νικόλαος 1054361

Γενικά σχόλια για το πρόβλημα :

Το πρόβλημα που αντιμετωπίζουμε στην παρούσα εργασία είναι ένα πολύ κοινό πρόβλημα και δεν είναι άλλο από την εύρεση μιας συνάρτησης η οποία περιγράφει ένα φαινόμενο όπου στην περίπτωσή μας δεδομένων κάποιων χαρακτηριστικών ενός κρασιού (είσοδος) πρέπει να βρούμε την αντίστοιχη αξιολόγηση (έξοδος) για αυτό . Μπορούμε να πούμε χάρην απλότητας ότι το σύστημα το οποίο πάμε να μοντελοποιήσουμε είναι στατικό αλλά δεν μπορούμε να αποφύγουμε την μη γραμμική του φύση . Στα πλαίσια της εργασίας θα εξετάσουμε διάφορους supervised (οδήγούμενους από δεδομένα) αλγορίθμους μηχανικής μάθησης . Συνεπώς θα προσεγγίσουμε το πρόβλημα με πιθανοτικές μεθόδους όπου η είσοδος-έξοδος μοντελοποιούντε ως τυχαίες μεταβλητές και σκοπός μας πια είναι να εκτιμήσουμε τις παραμέτρους ενός εκτιμήτη (συνάρτηση ή μετασχηματισμό τυχαίων μεταβλητών) βέλτιστου ως προς κάποια έννοια (π.χ. μέσο τετραγωνικό σφάλμα) . Επομένως το πρόβλημα ανάγεται σε ένα πρόβλημα βελτιστοποίησης όπου ως γνωστών σκοπός μας είναι η ελαχιστοποίηση της συνάρτηση κόστους μας μέσω του αλγορίθμου που θα χρησιμοποιήσουμε . Τέλος θα πρέπει να τονιστεί ότι το συγκεκριμένο πρόβλημα (και γενικά) μπορούμε να το δουμε είτε ως regression είτε ως classification (θα εξετάσουμε και τις δύο οπτικές) .

Τεχνικές περιγραφές και πειράματα με τους διάφορους αλγορίθμους:

Logistic Regression

Ο αλγόριθμος αυτός χρησιμοποιείται κυρίως για την μοντελοποίηση της πιθανότητας να υπάρχει μια κλάση ή ένα γεγονός.
Θεωρούμε το πρόβλημα ως πρόβλημα ταξινόμησης (classification) πολλαπλών κλάσεων, καθώς έχουμε ένα εύρος τιμών ποιότητας-κλάσεων από το 0 έως το 10. Ο αλγόριθμος ξεχωρίζει στην είσοδο του της εξαρτημένες μεταβλητές (wine quality στην περίπτωσή μας) και τις ανεξάρτητες μεταβλητές (όλα τα υπόλοιπα χαρακτηριστικά).
Η Sigmoid συνάρτηση εφαρμόζεται ως συνάρτηση κόστους στο Logistic Regrassion. Έτσι χρησιμοποιούμε αυτήν για την πρόβλεψη τιμών των πιθανοτήτων. Η μαθηματική της εξίσωση έχει ως εξής:

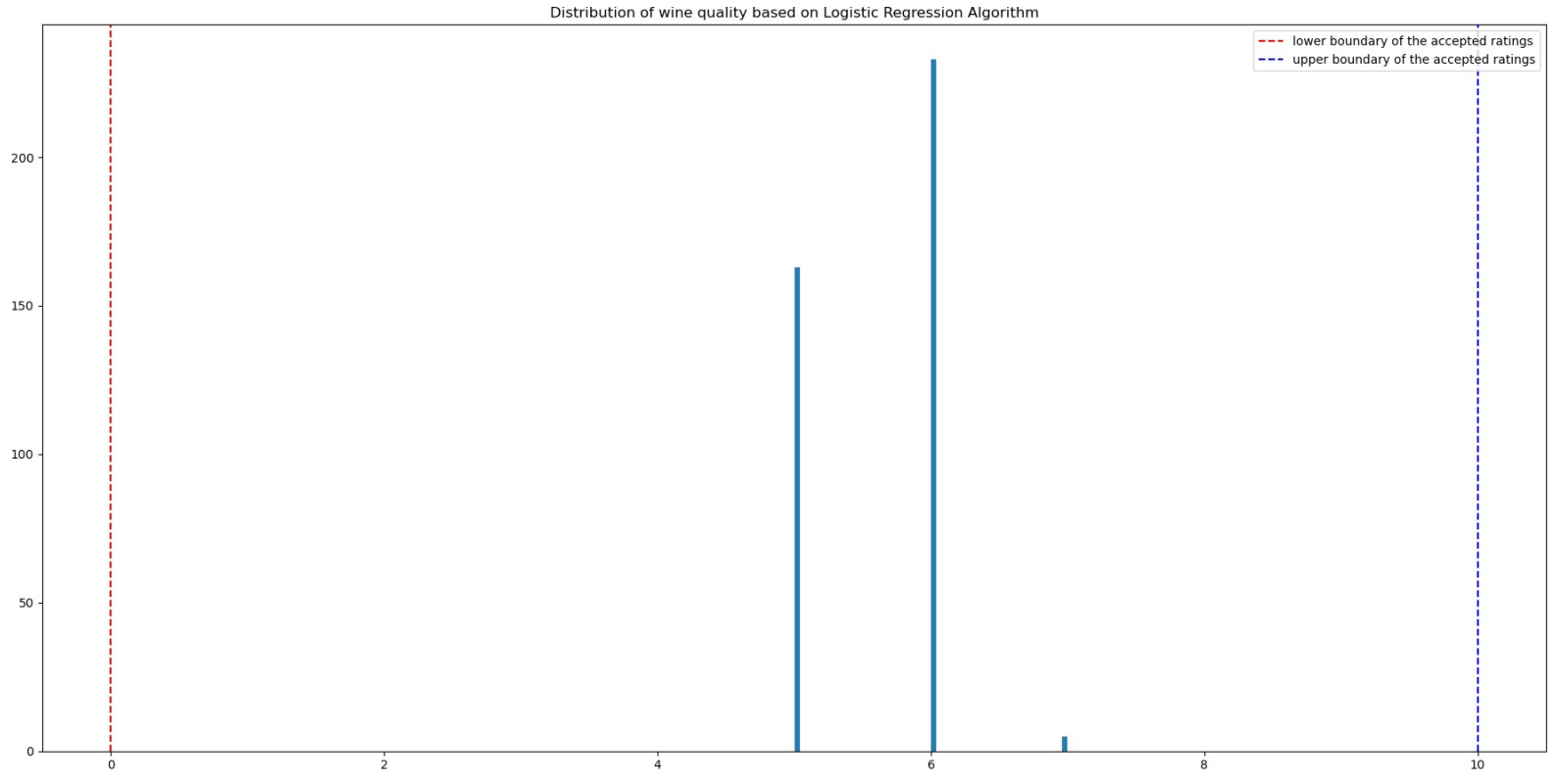
$$F(z) = \frac{1}{1-e^{-z}}$$

με $$z = w_0 + w_1.x_1 + w_2.x_2 + \cdots + w_n.x_n$$

όπου $w_0,\ldots,w_n$ αντιπροσωπεύει την παλινδρόμηση του συντελεστή του μοντέλου που αποκτάται μέσω του Maximum Likelihood Estimation, μεθόδου που προσεγγίζει τις παραμέτρους μιας πιθανοτικής κατανομής μεγιστοποιώντας την συνάρτηση πιθανότητας αποσκοπώντας στο να είναι τα αποτελέσματα των δεδομένων μας τα πιο πιθανά, και $x_0,\ldots,x_n$ αντιπροσωπεύει τις ανεξάρτητες μεταβλητές.

Βλέπουμε τρέχοντας τον αλγόριθμο με το training set,
χωρίζοντάς το στην συνέχεια σε train και test με αναλογία 75%-25%.
Επιπλέον υπολογίζουμε το mean square error καθώς και το validation του αλγορίθμου και το confusion matrix.

```
MSE:  0.5466666666666666
Testing validation : 0.61
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00         6
           5       0.63      0.75      0.68       122
           6       0.62      0.56      0.59       131
           7       0.53      0.51      0.52        35
           8       0.00      0.00      0.00         4

    accuracy                           0.61       300
   macro avg       0.30      0.30      0.30       300
weighted avg       0.59      0.61      0.60       300

[[ 0  0  1  1  0  0]
 [ 0  0  4  2  0  0]
 [ 0  0 92 28  2  0]
 [ 0  0 45 73 12  1]
 [ 0  0  5 12 18  0]
 [ 0  0  0  2  2  0]]
```



Decision Tree

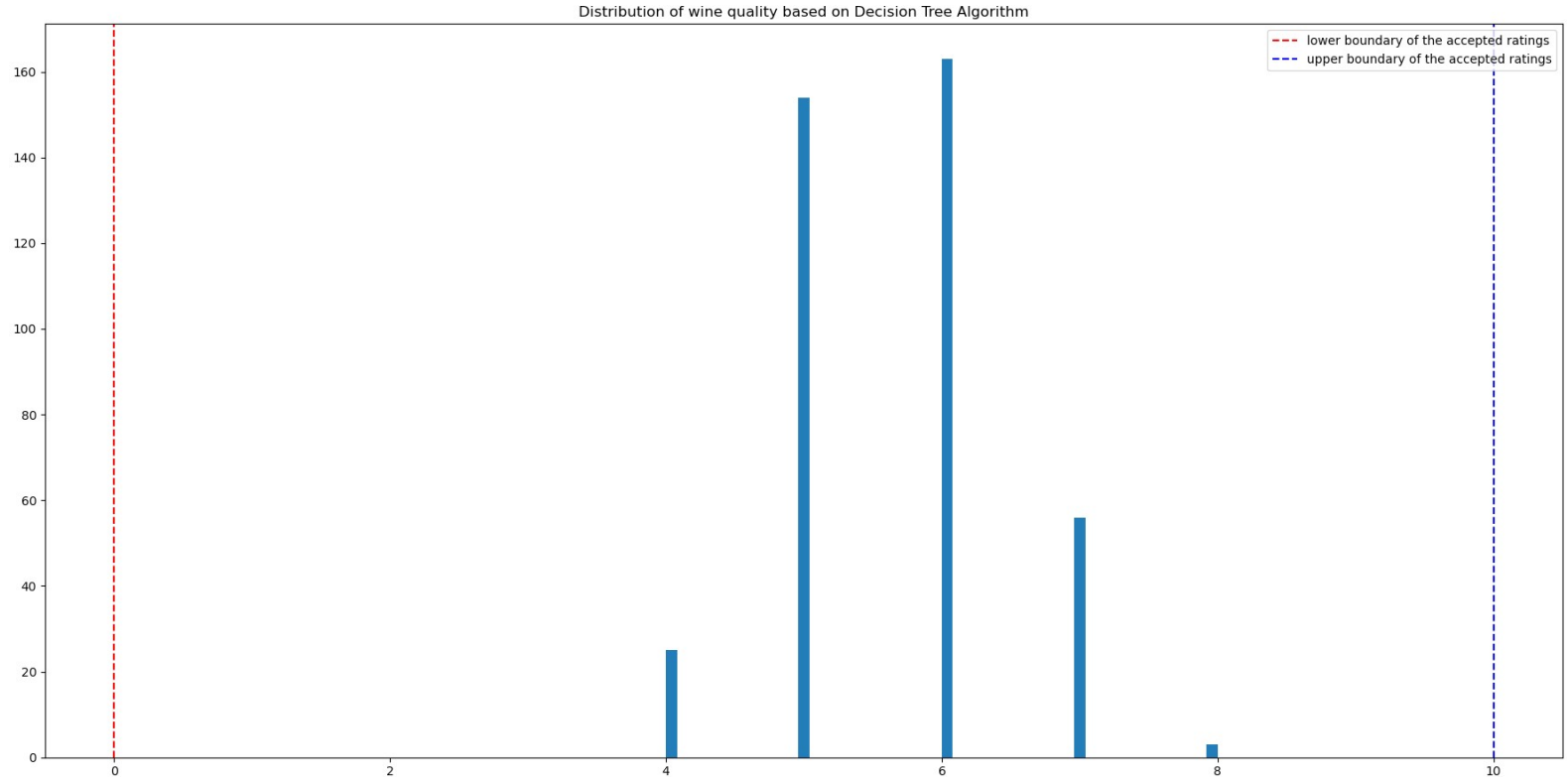
Το δέντρο αποφάσεων στην μηχανική μάθηση βασίζεται στις εξής αρχές: Λειτουργεί ως ένας classifier με δομή δέντρου που οι εσωτερικοί κόμβοι αντιπροσωπεύουν τα χαρακτηριστικά του dataset, οι διακλαδώσεις-ακμές απεικονίζουν τους κανόνες απόφασης και τα φύλλα δίνουν τα αποτελέσματα. Οι κόμβοι χωρίζονται σε κόμβους απόφασης και κόμβους φύλλα. Χρησιμοποιείται ο αλγόριθμος Classification and Regression Tree (CART). Εκτελείται ως εξής: Ξεκινώντας από τον κόμβο-ρίζα εισάγει όλο το dataset. Έπειτα βρίσκει το καλύτερο χαρακτηριστικό για να ξεκινήσει τον διαχωρισμό του dataset με την χρήση του Attribute Selection Measure. Στην συνέχεια διαιρεί το dataset σε subsets και δημιουργεί τον κόμβο απόφασης του καλύτερου χαρακτηριστικού. Τέλος επαναλαμβάνει αναδρομικά δημιουργώντας νέους κόμβους απόφασης έως ότου φτάσει στους κόμβους φύλλα για το classification μας (στην περίπτωσή μας το quality). Χρησιμοποιείται η μετρική Gini Index που δείχνει τον βαθμό της πιθανότητας να κατηγοριοποιηθεί λάθος μια τιμή εάν επιλεχθεί τυχαία.

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

Όπου p_i η πιθανότητα ενός αντικειμένου να κατηγοριοποιηθεί σε μια συγκεκριμένη κλάση. Προτιμάται για κόμβο-ρίζα το χαρακτηριστικό με το μικρότερο Gini Index.

Βλέπουμε τρέχοντας τον αλγόριθμο με το training set, χωρίζοντάς το στην συνέχεια σε train και test με αναλογία 75%-25%.
Επιπλέον υπολογίζουμε το mean square error καθώς και το validation του αλγορίθμου και το confusion matrix.

```
MSE:  0.6366666666666667
Testing validation : 0.6033333333333334
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00         6
           5       0.68      0.72      0.70       122
           6       0.66      0.53      0.59       131
           7       0.48      0.63      0.54        35
           8       0.14      0.25      0.18         4

    accuracy                           0.60       300
   macro avg       0.33      0.36      0.34       300
weighted avg       0.62      0.60      0.61       300

[[ 0  1  1  0  0  0]
 [ 0  0  3  2  1  0]
 [ 0  7 88 25  2  0]
 [ 2  1 34 70 19  5]
 [ 0  0  4  8 22  1]
 [ 0  0  0  1  2  1]]
```



Random Forest

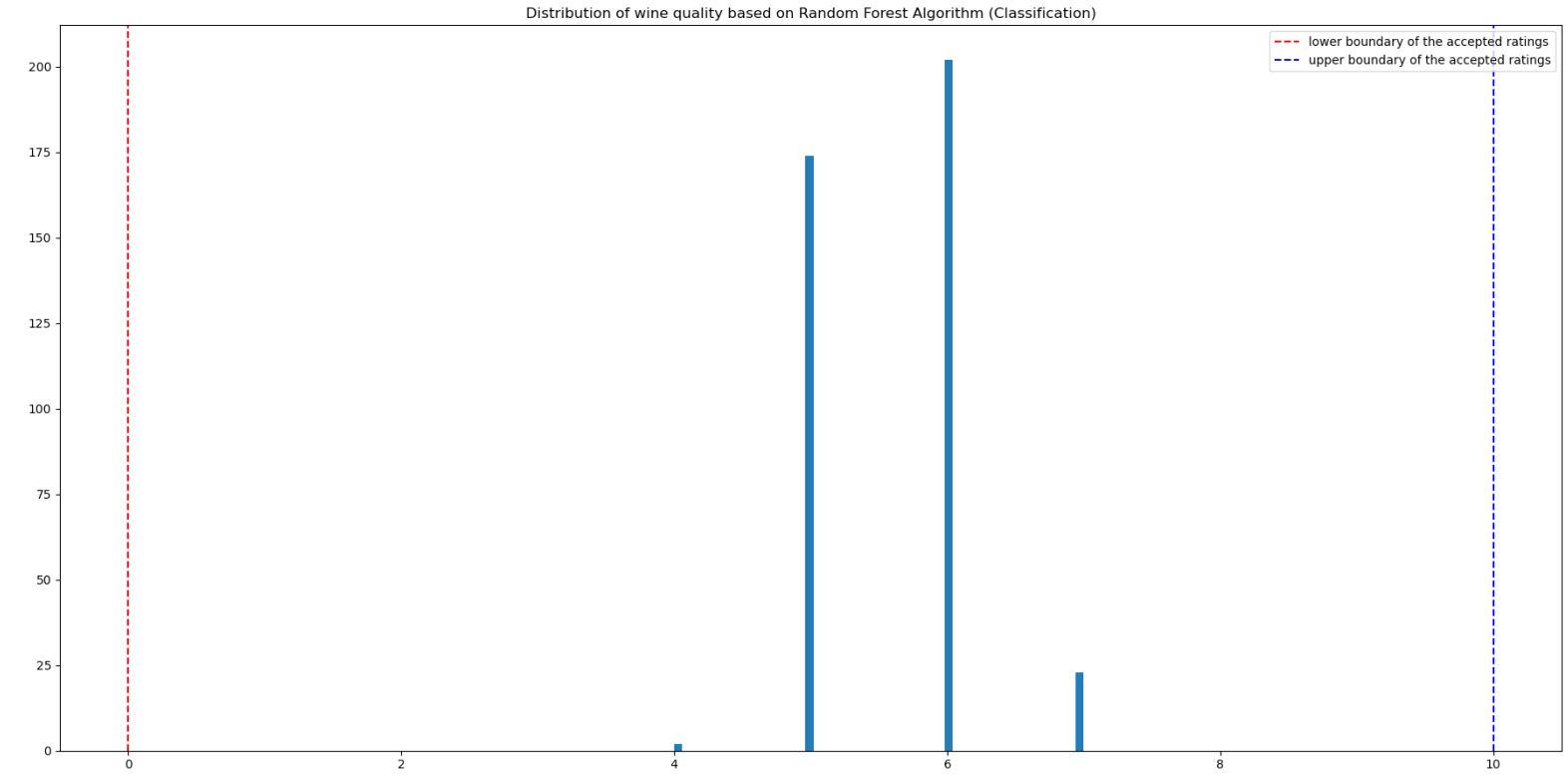
Ο αλγόριθμος Random Forest εκτελείται ως εξής: Επιλέγει έναν τυχαίο αριθμό N πλειάδων από το dataset. Δημιουργεί ένα δέντρο αποφάσεων βάσει αυτών των N πλειάδων. Εμείς επιλέγουμε τον αριθμό των δέντρων αποφάσεων που θέλουμε στον αλγόριθμό μας και επαναλαμβάνεται η παραπάνω διαδικασία. Στην περίπτωση που θεωρούμε το πρόβλημα τύπου Regression, για κάθε πλειάδα, κάθε δέντρο του δάσους προβλέπει μια τιμή για την έξοδο. Η τελική τιμή υπολογίζεται παίρνοντας τον μέσο όρο των τιμών από όλα τα δέντρα. Στην περίπτωση που θεωρούμε το πρόβλημα τύπου Classification, κάθε δέντρο προβλέπει προβλέπει την κατηγορία που θα ανήκει η πλειάδα. Έπειτα κάθε πλειάδα ανατίθεται στην κατηγορία που έχει την πλειοψηφία. Παρακάτω μελετάμε και τις 2 περιπτώσεις, Regression και Classification.

Επιπλέον θεωρούμε τον αριθμό των δέντρων (estimators) = 20 για τις δοκιμές μας.

Classification:

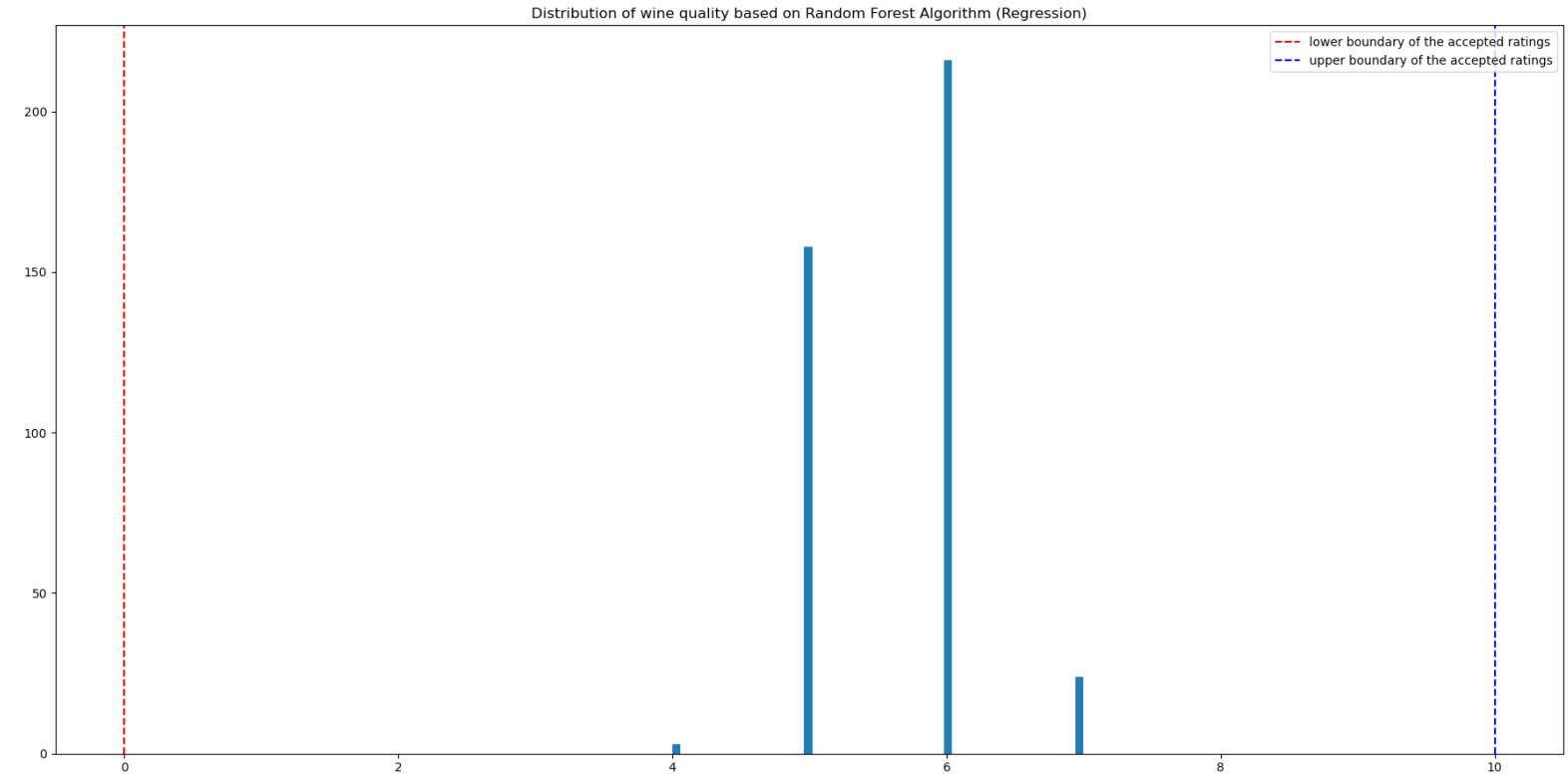
Βλέπουμε τρέχοντας τον αλγόριθμο με το training set, χωρίζοντάς το στην συνέχεια σε train και test με αναλογία 75%-25%. Επιπλέον υπολογίζουμε το mean square error καθώς και το validation του αλγορίθμου και το confusion matrix.

```
MSE:  0.4
Testing validation : 0.6766666666666666
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00         6
           5       0.68      0.83      0.75       122
           6       0.71      0.61      0.66       131
           7       0.57      0.60      0.58        35
           8       1.00      0.25      0.40         4

    accuracy                           0.68       300
   macro avg       0.49      0.38      0.40       300
weighted avg       0.67      0.68      0.66       300

[[  0   1   0   1   0   0]
 [  0   0   6   0   0   0]
 [  0   0 101  21   0   0]
 [  0   0  37  80  14   0]
 [  0   0   4  10  21   0]
 [  0   0   0   1   2   1]]
```



Regression:

```
MSE:  0.32559166666666667
Testing validation : 0.4680936995153475
```



- **Linear Regression**

Περιγραφή:

Μια από τις βασικές υποθέσεις που γίνονται στην μέθοδο αυτή είναι ότι η υπο-συνθήκη μέση τιμή της εξαρτημένης μεταβλητής Y και τα δεδομένα X (ανεξάρτητη μεταβλητή) σχετίζονται με έναν γραμμικό μετασχηματισμό $E(Y|X)=f(X)$ (η f είναι affine συνάρτηση) και επίσης η f δεν είναι στοχαστική αλλά ντετερμινιστική . Επομένως εξ ορισμού πρακτικά αυτό που καλούμαστε να λύσουμε είναι ένα γραμμικό σύστημα εξισώσεων $Ax=b$ όπου το A (design matrix) περιέχει ως στήλες τις διάφορες τυχαίες μεταβλητές (ανεξάρτητες μεταβλητές ή predictors → χαρακτηριστικά του κρασιού) , x είναι οι παράμετρες του εκτιμήτη που ψάχνουμε και y είναι η εξαρτημένη μεταβλητή (αξιολόγηση κρασιού) . Το γραμμικό σύστημα εξισώσεων το οποίο δημιουργείται είναι είναι ένα underdetermined σύστημα (έχουμε πιο πολλες εξισώσεις παρά αγνώστους tall πίνακας A) επομένως το πιο πιθανό είναι να μην έχει λύση (έχει λύση ανν το b ανήκει στο range(A) (κάπως απίθανο)) . Επομένως αυτό που κάνουμε είναι να προσεγγίζουμε το b προβάλοντάς το στο χώρο στηλών του A . Πιο συγκεκριμένα λύνουμε ένα άλλο σύστημα $A^TAx=A^Tb$ όπου τώρα το μητρώο A^TA είναι τετραγωνικό συμμετρικό και θετικά ορισμένο και ελπίζουμε ότι αντιστρέφεται (υπάρχει και περίπτωση να μην είναι αντιστρέψιμο εάν το A δεν είναι full rank *) . Η λύση η οποία προκύπτει μετα την αντιστροφή είναι κλειστής μορφής,μοναδική (γιατί στην ουσία είναι το ελάχιστο μιας κυρτής συνάρτησης που είναι μοναδικό) και είναι η $x^{opt}=(A^TA)^{-1}A^Tb$ και είναι λύση η οποία είναι βέλτιστη ως προς το μέσο τετραγωνικό σφάλμα (καθώς όπως προαναφέραμε η ελαχιστοποίηση της συναράρτησης κόστους (τετραγωνικό σφάλμα) είναι convex πρόβλημα) . Η προηγούμενη μέθοδος υπολογισμού του βέλτιστου εκτιμητή είναι γνωστο εργαλείο της γραμμικής αλγεβρας και λέγεται Least Squares . Τέλος πρέπει να τονίσουμε ότι για να

εκτελέσει κάποιος linear Regression θα πρέπει να λάβει υπόψιν του και την γραμμική εξάρτηση ή το corellation (αφού μιλάμε για τ.μ.) των εξαρτημένων μεταβλητών καθώς εάν υπάρχει τότε δεν θα μπορεί να αντιστραφεί ούτε ο $A^T A$ (μεγάλος δείκτης κατάστασης μια ιδιοτιμή κοντά στο μηδέν) με αποτέλεσμα η ακρίβεια των αποτελεσμάτων να είναι μειωμένη (numerical instability) . Επομένως σε τέτεια περίπτωση θα έπρεπε να εξεταστεί κάποια τεχνική Regularization της λύσης (L2 Ridge ,L1 Lasso) .

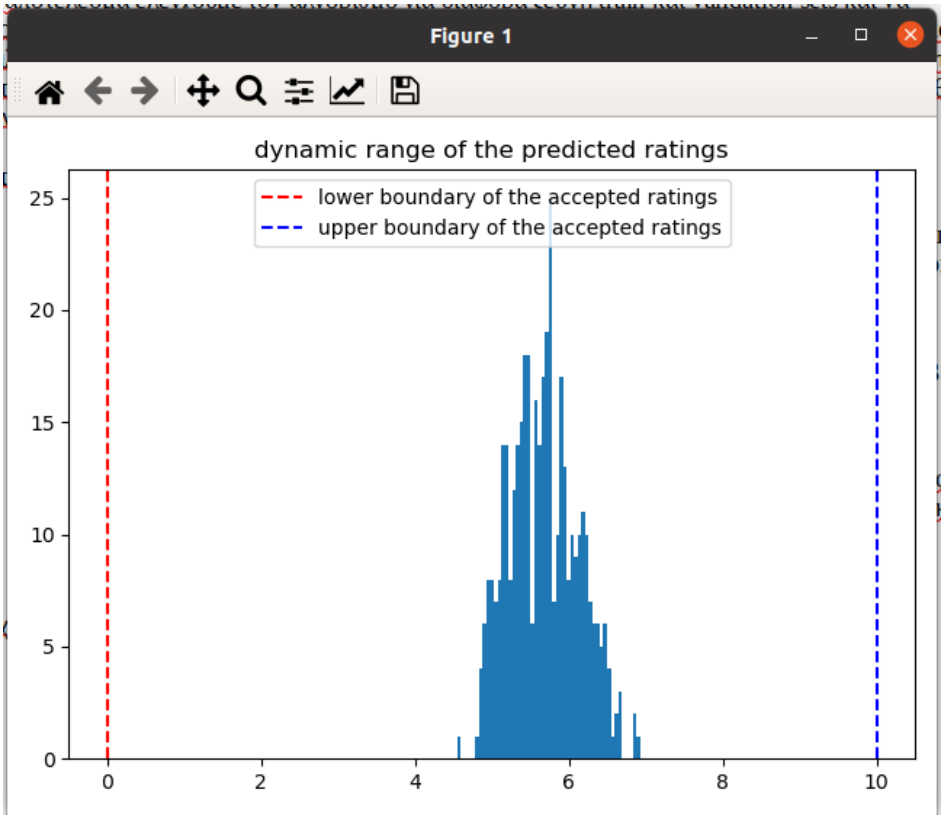
Χρήση K-fold cross validation :
Η τεχνική k-fold cv είναι ευρέως χρησιμοποιούμενη στο evaluation αλγορίθμων machine learning (και ειδικά supervised) . Πιο συγκεκριμένα αυτό που κάνει είναι επαναληπτικά να χωρίζει το αρχικό train dataset (αυτό που περιέχει και τις labeled εξόδους) σε train και validation sets . Σε κάθε επανάληψη επιλέγει ένα σημείο στο οποίο θα χωριστεί το dataset ,έπειτα κάνει train και validate το μοντέλο στα αντιστοιχα sets που δημιουργήθηκαν και τελικά κρατάει τα στατιστικά για το training και το validation του τρέχοντος fold (επανάληψης) . Η προηγούμενη διαδικασία επαναλαμβάνεται με τη μόνη διαφορά ότι επιλέγεται διαφορετικό σημείο διαχωρισμού κάθε φορά με αποτέλεσμα ελέγχουμε τον αλγόριθμο για διάφορα ζεύγη train και validation sets και να πέρνουμε έναν μέσο όρο αυτών των στατιστικών για κάθε fold . Προφανώς εδώ παίζει ρόλο και η επιλογή του k (όσο το μεγαλώνεις τόσο το train set θα μεγαλώνει σε πλήθος και το validation το αντίθετο (#train + #validation=#συνολικού dataset για train) (ακραίες περιπτώσεις : k=2 τότε #train=#validation=#συνολικού / 2 , k=#συνολικού τότε #train=#συνολικού #validation=0) .

Αποτελέσματα πειραμάτων του αλγορίθμου:

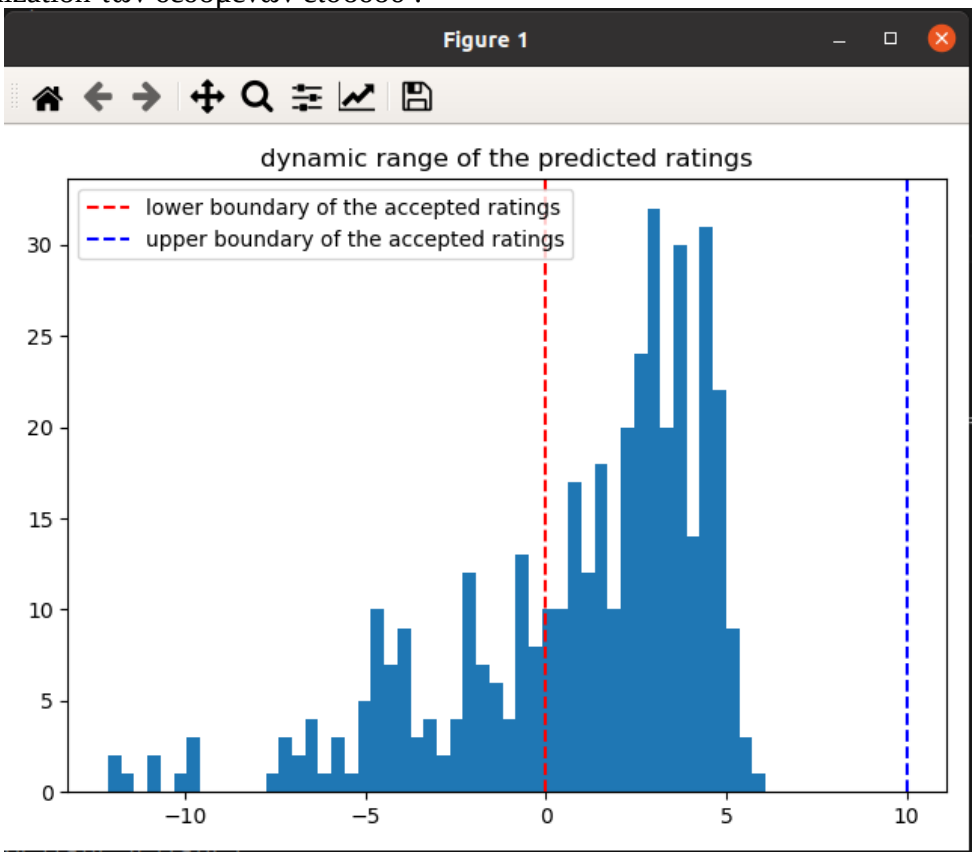
- Για κάθε fold (k=5) αυτό που κάναμε είναι train του μοντέλου (Least Squares) στο train set (βρήκαμε τον εκτιμητή) , στη συνέχεια πήραμε τις εξόδους του μοντέλου στο validation set (με ένα απλό Matrix vector multiplication) και τέλος υπολογίσαμε το MSE για το validation αφού γνωρίζουμε τις εξόδους (το MSE ορίζεται ως $\|y_{pred} - y_{true}\|_2^2 \Big/ number\ of\ samples$) .Για όλα τα validations ο μέσος όρος βγήκε 0.435 .

- Στη συνέχεια εκπαιδεύσαμε (Least Squares) το μοντέλο με όλα τα δεδομένα αυτήν τη φορά (δεν κάναμε k-fold CV δηλαδή) , πήραμε τις εξόδους με τον ίδιο τρόπο όπως και πρίν και τέλος ελέγξαμε το δυναμικό εύρος των αποκρίσεων του μοντέλου μας στις άγνωστες ειδόδους που δίνονται (test set) .

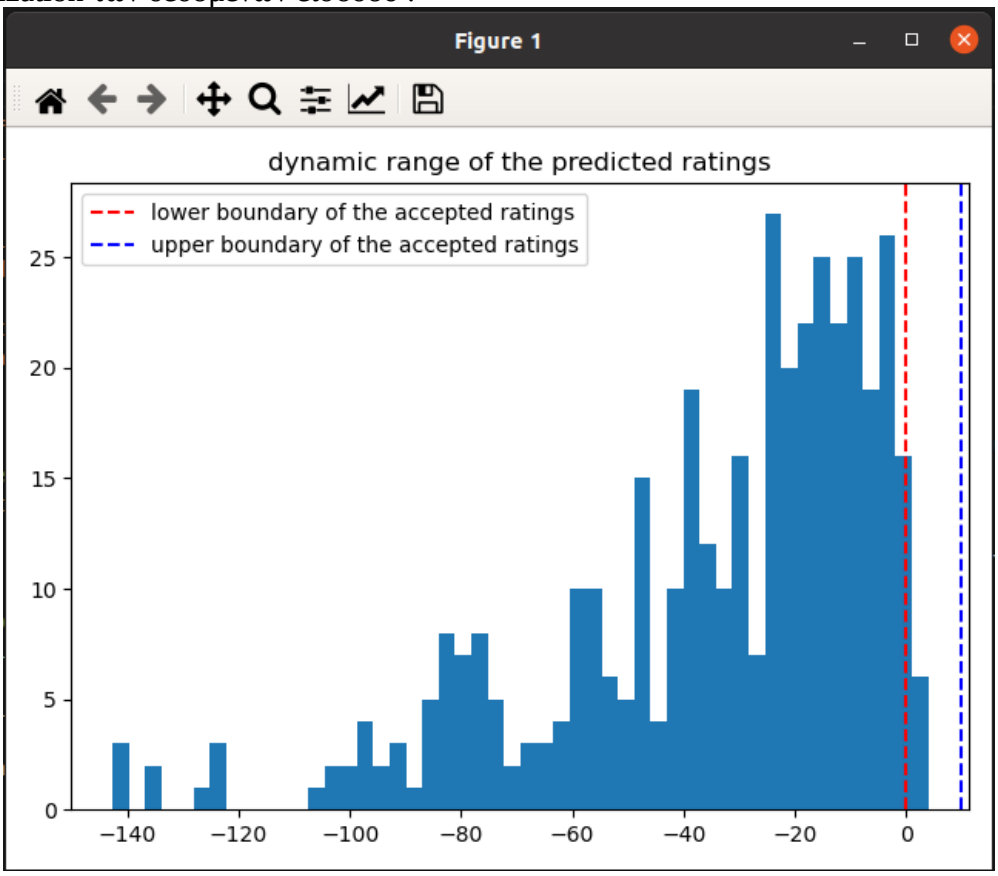
Δίχως standardization των δεδομένων εισόδου :



Με standardization των δεδομένων εισόδου :



Με normalization των δεδομένων εισόδου :



- Όπ
ου

standardisation είναι η αφαίρεση της μέσης τιμής από κάθε τυχαία μεταβλητή (στήλες του A) και στη συνέχεια η διαίρεση με την τυπική απόκλιση . Αποτελεί ένα είδος preproccesing των train δεδομένων εισόδου του μοντέλου όπου σκοπός είναι τα χαρακτηριστικά μας να έρθουν στο ίδιο scale . Δίνεται:

$$X' = \frac{X - \mu}{\sigma}$$

- Όπου **normalization** είναι το scaling των δεδομένων μου στο εύρος [0,1] (υπολογίζεται με την ίδια διαδικασία όπως και πρίν σε κάθε στήλη) και δίνεται από τον τύπο:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Παρατηρούμε ότι οι προβλέψεις χωρίς το standardisation είναι πιο κοντά στην πραγματικότητα άρα αυτές οι δύο τεχνικές preproccesing τελικά δεν βοηθάνε και πολύ στο πρόβλημά μας . Μια πιθανή αιτιολόγηση είναι ότι (έπειτα από παρατήρηση των ιστογραμμάτων) οι ανεξάρτητες μεταβλητές μας (οι περισσότερες) δεν ακολουθούν gaussian κατανομή και επομένως δεν μας βοηθάει το standardization (αφού και οι μετασχηματισμένες μεταβλητές πάλι δεν θα ακολουθούν gaussian κατανομή).

K-nearest neighbor (custom):

Ο αλγόριθμος υλοποιήθηκε από εμάς και δεν είναι ακριβώς η υλοποίηση της python (ωστόσο τη συγκρίναμε με την built-in της sklearn) . Περιγράφεται παρακάτω.

K_nn_custom(k , querry_wine , train_data , target_data)

1. Κανονικοποίησε το query_wine και κάθε γραμμή (example) του train_data (διαίρεση με την L2 νόρμα του αντίστοιχα) .
2. Πάρε όλα τα εσωτερικά γινόμενα του querry με τον πίνακα train_data (ένα Matrix_vector Multiplication) . Το διάνυσμα εξόδου αποτελέι όλα τα similarity coeficients (εύρος [-1,1]) του query_wine με όλα τα examples .
3. Κάνουμε sort του vector με τα coefs που προέκυψε και πέρνουμε τα indicies των top k αποκρίσεων (μεγαλύτερα similarities του similarity vector) . Έπειτα δεικτοδοτούμε το target _data όπου πέρνουμε τα ratings αυτών των top k όμοιων με το query_wine .
4. Πέρνουμε το πιο συχνά εμφανιζόμενο rating από τα προηγούμενα (με τη βοήθεια του ιστογράμματος) .

Βασική διαφορά με τον K-nn είναι ότι σε αυτόν η εύρεση των top k όμοιων γίνεται με βάση την μετρική της ευκλείδιας απόστασης ενώ εμείς χρησιμοπιούμε εσωτερικά γινόμενα . Επίσης καθοριστικής σημασίας βήμα είναι το 1ο (αφού χρησιμοποιούμε εσωτερικά γινόμενα) καθώς με αυτόν τον τρόπο γινόμαστε invariant στο scale των διανυσμάτων που συγκρίνουμε (έυρος απόκρισης [-1,1]) . Τέλος θα πρέπει να αναφέρουμε ότι ¨λογικά¨ θα πρέπει να πέρνουμε παρόμοια απότελέσματα και με την L2 απόσταση αν και τώρα το εύρος θα είναι [0,2] (εάν κανονικοποιήσουμε πάντα τα διανύσματα) και όσο μικρότερο τόσο το καλύτερο .

Αποτελέσματα πειραμάτων του αλγορίθμου k-nn :

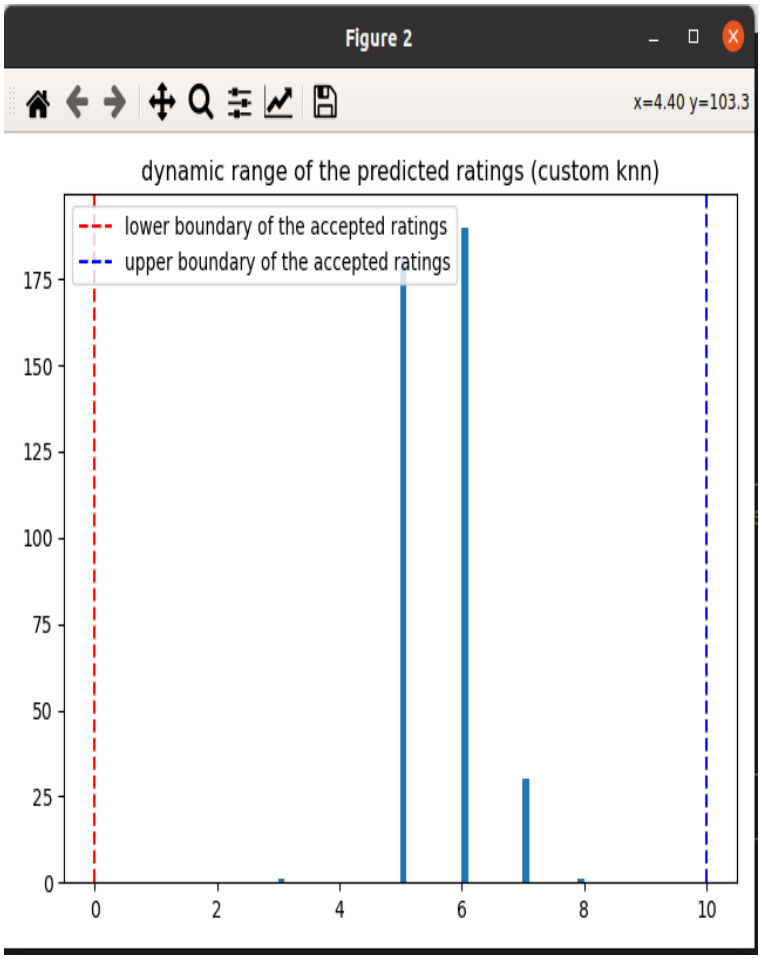
Για το evaluation του αλγορίθμου χρησιμοποίθηκε και εδώ η τεχνική Cross validation για k = 5 με μετρική το MSE . Πρέπει να σημειωθεί ότι εκτελέσαμε τις ίδιες διαδικασίες (5-fold CV) και για τον built in αλγόριθμο της sklearn (με την ίδια παράμετρο k εννοείται) στον οποίο η μετρική που χρησιμοποιήθηκε για τον καθορισμό των top k γειτόνων είναι η L2 απόσταση .

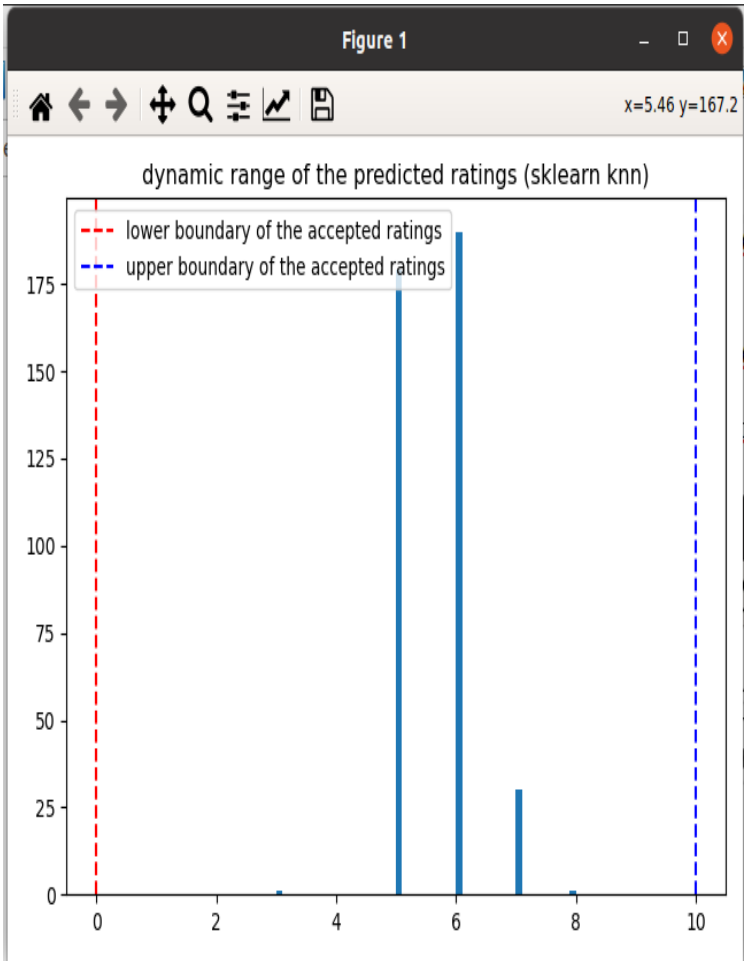
- **k = 5**

1)**Με κανονικοποίηση των διανυσμάτων εισόδου και εξόδου (κάνοντάς τα μοναδιαία)**
The mse and accuracy obtained from the average of all the fold validations (for the two algorithms) is:
MSE:
0.8806938633193863 (custom k-nn)
0.8806938633193863 (sklearn k-nn)

Accuracy:
0.48578451882845186 (custom k-nn)
0.48578451882845186 (sklearn k-nn)

Εύρος αποκρίσεων στα άγνωστα δεδομένα :





Παρατηρούμε ότι το cosine similarity (αφού κανονικοποιήσαμε) δίνει τα ίδια αποτελέσματα με την L2 απόσταση . Γινόμαστε invariant στο magnitude (L2 στάθμη) των δεδομένων και εξαρτόμαστε μόνο από τη γωνία τους καθώς όλα δεδομένα βρίσκονται τώρα στον ίδιο κύκλο. Για να προχωρήσουμε σε κανονικοποιήση των δεδομένων κάνουμε την υπόθεση ότι το magnitude των δεδομένων δεν παίζει κάποιο ρόλο ή εισάγει κάποιου είδους θόρυβο οπότε και τον αφαιρούμε .

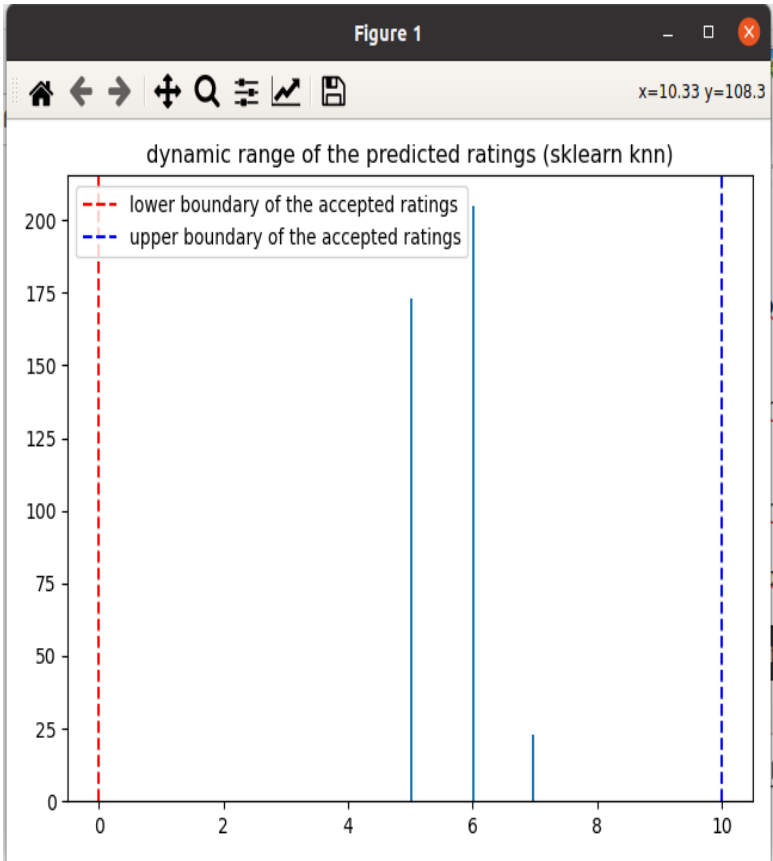
2)**Χωρίς κανονικοποίηση των διανυσμάτων εισόδου και εξόδου (raw data)**

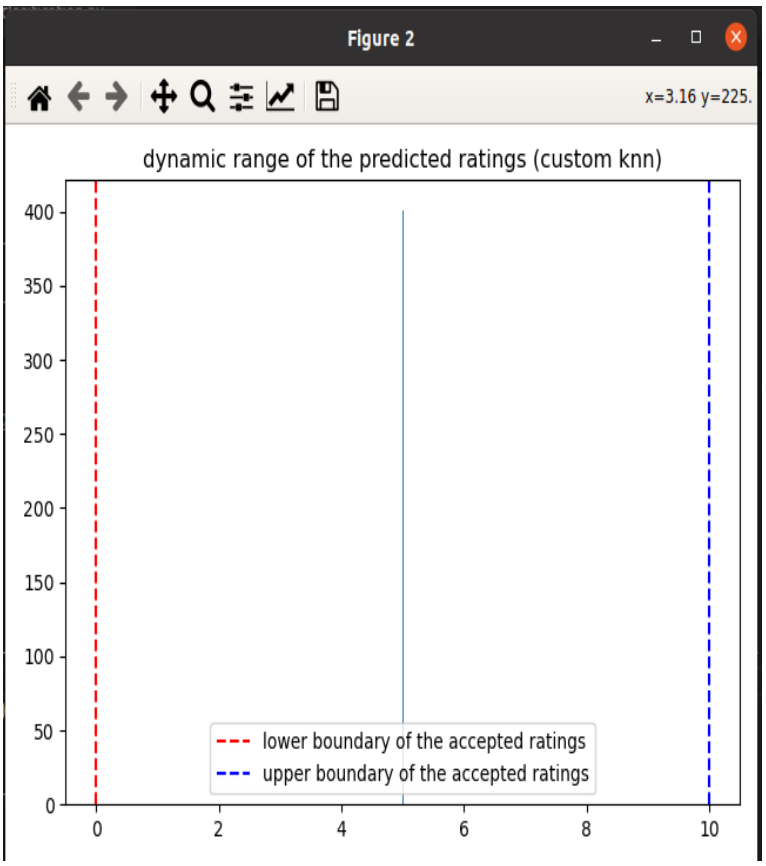
The mse and accuracy obtained from the average of all the fold validations (for the two algorithms) is:

MSE:
1.0972559274755926 (custom k-nn)
0.799700139470014 (sklearn k-nn)

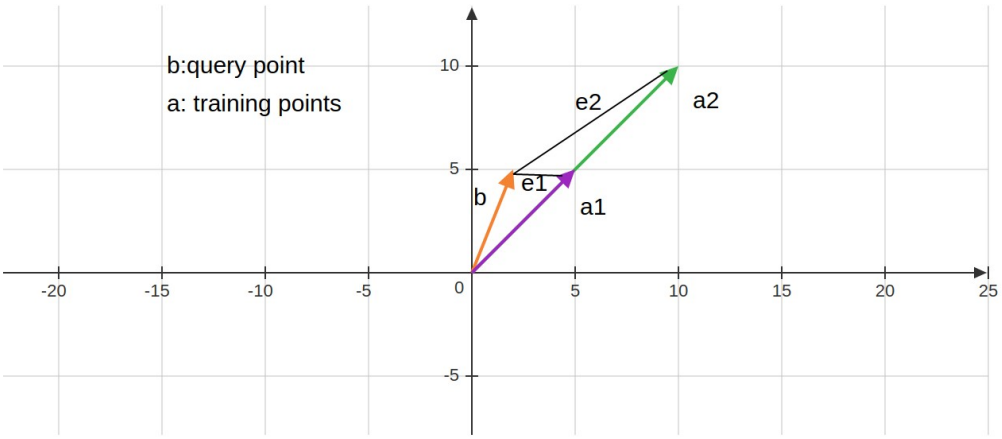
Accuracy:
0.42807880055788006 (custom k-nn)
0.47911785216178526 (sklearn k-nn)

Εύρος αποκρίσεων στα άγνωστα δεδομένα :





Παρατηρούμε ότι στα raw δεδομένα (χωρίς preprocessing) ο αλγόριθμος που υλοποιήσαμε έχει χειρότερες μετρικές σε σχέση με παραπάνω (με preprocessing) και αυτό φαίνεται και από τις αποκρίσεις στα άγνωστα δεδομένα . Ο αλγόριθμος αδυνατεί να ταξινομήσει τα πρότυπα εισόδου στις διάφορες κλάσεις και τα ταξινομεί όλα στην κλάση 5 . Έπειτα από debugging παρατηρήσαμε ότι για κάθε ένα query_wine οι top k γείτονες του training set ήταν οι ίδιοι το οποίο έγκειται στο γεγονός ότι για αυτούς τους γείτονες το dot product (για κάθε query wine) έδινε τη μεγαλύτερη απόκριση διότι το μέτρο τους (των top k) ήταν λογικά πολύ μεγαλύτερο σε σχέση με όλα τα υπόλοιπα . $a^T b = cos\theta_0 * \|\alpha\| * \|b_0\| = f(\|\alpha\|)$ Η προηγούμενη σχέση δείχνει ότι αν χρησιμοποιήσουμε ως μετρική για τον καθορισμό των γειτόνων το dot product τότε (αφού το μέτρο του b είναι σταθερό) η έξοδος της μετρικής θα εξαρτιέται από το magnitude των των δεδομένων του dataset και επομένως το max (αυτών των εξόδων) θα πολώνεται προς εκείνα τα training δεδομένα που έχουν υψηλό magnitude (αν υποθέσουμε βέβαια και ότι αυτά τα α βρίσκονται στην ίδια διεύθυνση και σε κάποιο βαθμό η γωνία είναι κοντά στο 1) . Από την άλλη η L2 απόσταση είναι θα λέγαμε invariant στο magnitude διότι για να αποφασίσουμε για κάποιον από τους top k γείτονες επιλέγουμε το min της απόκρισης της μετρικής . Το ακόλουθο γράφημα δείχνει την ενδεχόμενη κατάσταση :



L2 απόσταση : $\|e_1\|_2 < \|e_2\|_2$ επομένως ο αλγόριθμος θα αποφασίσει α1 (σωστό)
dot products : $a_1^T b < a_2^T b$ επομένως ο αλγόριθμος θα αποφασίσει το α2 (λάθος)

Τέλος πρέπει να τονίσουμε ότι ο αλγόριθμος k-nn έχει το θετικό ότι είναι απλός με αποτέλεσμα να μπορούμε να αναλύσουμε ευκολότερα τη συμπεριφορά του (όπως παραπάνω). Ωστόσο όπως είδαμε δεν είναι robust καθώς η απόδοσή του εξαρτάται πολύ από την ποιότητα του dataset που διαθέτουμε και για μικρές διαταραχές στην είσοδό του ενδέχεται να παράξει πολυ διαφορετικά αποτελέσματα . Επίσης το κόστος του (στη γενική περίπτωση όχι στο πρόβλημά μας) σε χώρο αλλά και σε χρόνο είναι μεγάλο και εξαρτάται από το μέγεθος του προβλήματος (dimensionality εισόδου) .

Feedforward Nueral Net (multilayer perceptron with one hidden layer) :

Περιγραφή :

Γνωρίζοντας ότι το πρόβλημα έχει μη γραμμική φύση θελήσαμε να εξετάσουμε και αλγορίθμους όπως τα Neural Nets τα οποία ξέρουμε ότι μπορούν να αντιμετωπίσουν τέτοιού είδους προβλήματα από το Universal aproximation theorem . Ως εκ τούτου Θα αποπειραθούμε να εκπαιδεύσουμε 2 (ένα για το regression πρόβλημα και ένα για το classification) multilayer perceptrons με ένα κρυφό επίπεδο το καθένα . Γενικά τα Neural Nets έχουν κάποιες learnable παραμέτρους (τα βάρη του δικτύου) και κάποιες υπερπαραμέτρους (π.χ. πλήθος κόμβων κρυφού επιπέδου , βήμα μάθησης , batch size) . Επίσης πρέπει να τονιστεί ότι προφανώς η λύση (βέλτιστες παράμετροι) που ελαχιστοποιούν τη συνάρτηση κόστους ως προς τις παραμέτρους του δεν δίνεται από κλειστό (όπως π.χ. linear regression) καθώς με την εισαγωγή μη γραμμηκοτήτων (συναρτήσεις ενεργοποίησης) η συνάρτηση κόστους γίνεται non convex με αποτέλεσμα να μην έχει ένα ολικό ελάχιστο . Η έξοδος ενός MLP είναι μια σύνθεση συναρτήσεων και η εξίσωση ενός κόμβου i σε ένα επίπεδο j είναι :
Πρέπει να αναφέρουμε ότι η ελαχιστοποίηση της συνάρτησης κόστους γίνεται σε εποχές δηλαδή γίνεται με μια επαναληπτική διαδικασία (gradient decsent) ενώ τα gradients υπολογίζονται με τον αλγόριθμο backpropagation . Ως κριτήριο τερματισμού ορίζουμε είτε κάποιο πλήθος εποχών είτε μια early stopping διαδικασία . Τέλος ένα Neural Net μπορεί να εκπαιδευτεί είτε “μαζικά ”(batch learning) είτε “on-line” ένα παράδειγμα τη φορά είτε (ενδιάμεση λύση) σε mini-batches .

- batch learning (steepest descent)

Τροφοδοτούμε το δίκτυο με όλα τα παραδείγματα (δεδομένα) που διαθέτουμε , υπολογίζουμε το loss δια το πλήθος των παραδειγμάτων → έκτελούμε τον backpropagation για να κάνουμε refinement των συναπτικών βαρών → επόμενη εποχή εκπαίδευσης και επανάληψη των προηγούμενων βημάτων
+ Εγγυάται την εύρεση ενός τοπικού ακροτάτου (αφού η εκτίμηση του διανύσματος κλίσης της συνάρτησης κόστους είναι ακρίβής (βασίζεται σε όλα τα δεδομένα))
- Μεγάλες απαιτήσεις σε μνήμη εάν τα δεδομένα είναι πολλά και μεγάλης διάστασης (μη πρακτικό)

- on-line learning (LMS)

Τροφοδότηση του δικτύου με ένα παράδειγμα → υπολογισμός του loss → εκτέλεση backpropagation και refinement συναπτικών βαρών (σύμφωνα μόνο με το loss του τρέχοντος παραδείγματος) → εκτέλεση των προηγούμενων βημάτων για το επόμενο παράδειγμα μέχρι να τελειώσουν → επόμενη εποχή εκπαίδευσης και επανάληψη των προηγούμενων βημάτων

+Μπορεί να υλοποιηθεί στην πράξη λόγω της απλότητάς του και του μικρού υπολογιστικού κόστους και μπορεί να αποφύγει την παγίδευση σε τοπικό ελάχιστο .

-Ο υπολογισμός του gradient βασίζεται σε ένα παράδειγμα επομένως μπορεί να περιέχει θόρυβο.

Εμείς θα ακολουθήσουμε την ενδιάμεση λύση η οποία είναι η εκπαίδευση σε mini-batches (ένας συνδιασμός των παραπάνω τεχνικών για να πάρουμε όσο το δυνατόν περισσότερα θετικά) . Πιο συγκεκριμένα :
Τροφοδότηση του δικτύου με ένα mini-batch (υποσύνολο παραδειγμάτων) → υπολογίζουμε το loss δια το πλήθος των παραδειγμάτων του mini-batch → εκτέλεση backpropagation και refinement συναπτικών βαρών → εκτέλεση των προηγούμενων βημάτων για το επόμενο mini-batch μέχρι να τελειώσουν → επόμενη εποχή εκπαίδευσης και επανάληψη των προηγούμενων βημάτων

Τα απαραίτητα που πρέπει να ορίσουμε έτσι ώστε να εκπαιδεύσουμε ένα Neural Net είναι τα ακόλουθα :

1. **Συνάρτηση Κόστους**

- Για το regression δίκτυο : $J(\underline{w}) = \frac{1}{\mathrm{N}} * \|\underline{\hat{y}} - \underline{y}\|_2^2$ (MSE)

- Για το classification δίκτυο : $J(\underline{w}) = \sum_{i=1}^{N} -\log\left(g(\hat{y}[true\,class])_i\right)$ (cross entropy loss)

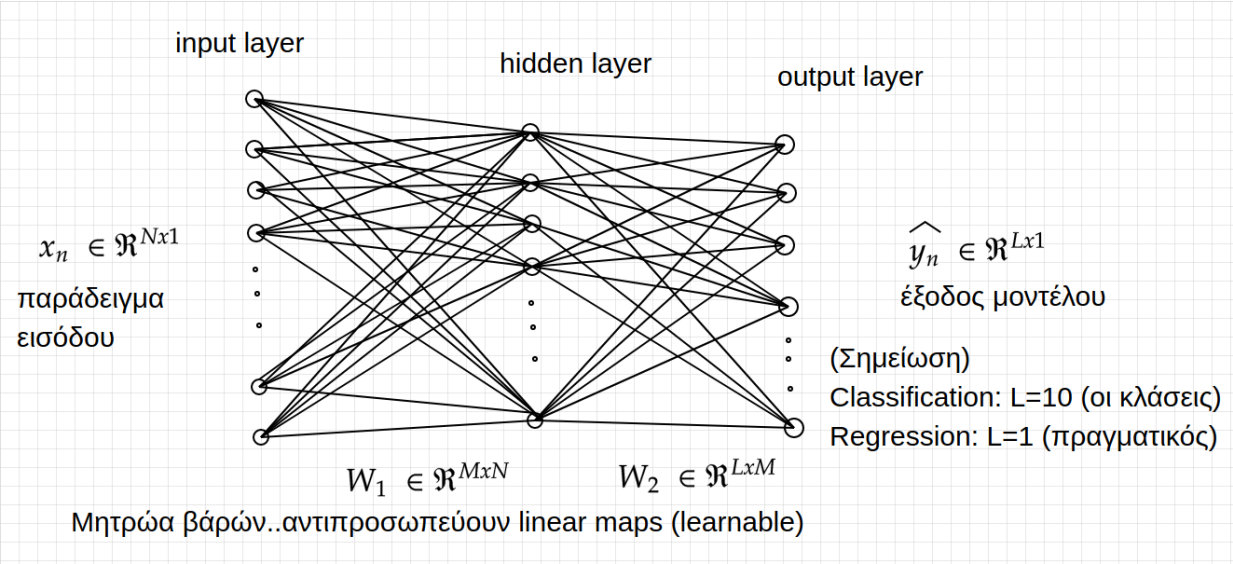
(τα προηγούμενα είναι για ένα mini-batch!)

με $g(\hat{y}[class]) = \frac{e^{\hat{y}[class]}}{\sum_{j}^{\#\text{κλάσεων}} e^{\hat{y_j}}}$ λέγεται softmax συνάρτηση και εφαρμόζεται στο διάνυσμα εξόδου $\hat{y_j}[.]$ το οποίο και μετατρέπει σε κατανομή (είναι η επέκταση της λογιστικής σε πολλές κλάσεις) . Έτσι το i-στο στοιχείο του διανύσματος g που προκύπτει μας δείχνει την πιθανότητα την οποία εκτιμά το δίκτυο η είσοδος να ανήκει στην κλάση i (δηλαδή η συνάρτηση g μοντελοποιεί την pdf της εξόδου) .

- $\underline{w}$ το διάνυσμα βαρών του δικτύου
- $\underline{\hat{y}}$ η έξοδος του δικτύου και $\underline{y}$ η πραγματική έξοδος (και τα δύο αφορούν το mini-batch) .
- N είναι το mini - batch size που χρησιμοποιήσαμε

(Η εξάρτηση από το διάνυσμα βάρους δεν φαίνεται ξεκάθαρα από τους παραπάνω τύπους ωστόσο μπορεί να φανεί με παιρετέρω ανάλυση των τύπων και ειδικά της εξόδου του δικτύου).

2. **Αρχιτεκτονική Δικτύου**



Να σημειωθεί ότι το N=11 είναι η διάσταση του input vector χαρακτηριστικών (δεν έχει σχέση με το παραπάνω N!!) . Και επίσης ότι εφαρμόστηκε μια συνάρτηση ενεργοποίησης στην έξοδο του κρυφού επιπέδου (η Relu) και των δύο προβλημάτων (regression , classification). (η Relu μηδενίζει τις αρνητικές αποκρίσεις και αφήνει αναλοίωτες τις θετικές) .

Πειράματα (Pytorch):

Αρχικά πρέπει να πούμε ότι :

1. Για το validation των αλγορίθμων χρησιμοποιήσαμε και πάλι k-fold CV (k=5) για κάποιες υπερπαραμέτρους οι οποίες έπειτα από πειραματισμό ‘παγώθηκαν’.
2. Για τον τερματισμό του training χρησιμοποιήσαμε τα 2 ακόλουθα κριτήρια

- Μέγιστο αριθμό εποχών 100 .
- Χρησιμοποιήσαμε και early-stopping κατά το οποίο σε κάθε εποχή κρατάμε το validation loss (το οποίο είναι το loss στο υποσύνολο δεδομένων 20% τα οποία δεν έχουμε χρησιμοποιήσει για εκπαίδευση) και εάν αυτό σταματήσει να μειώνεται για κάποιον προκαθορισμένο αριθμό απο epochs (patience) τότε σταματάμε την εκπαίδευση .

3. Αν κάποιες ο αλγόριθμοι βγάλουν λίγο διαφορετικές προβλέψεις στο testing phase από αυτές που φαίνονται στα πειράματα παρακάτω είναι “λογικά” λόγω του αλγορίθμου **στοχαστικής** κατάβασης που χρησιμοποιείται για το refinement των βαρών (συγκλίνει ως προς τη μέση τιμή στις παραμέτρους παραμέτρους) .

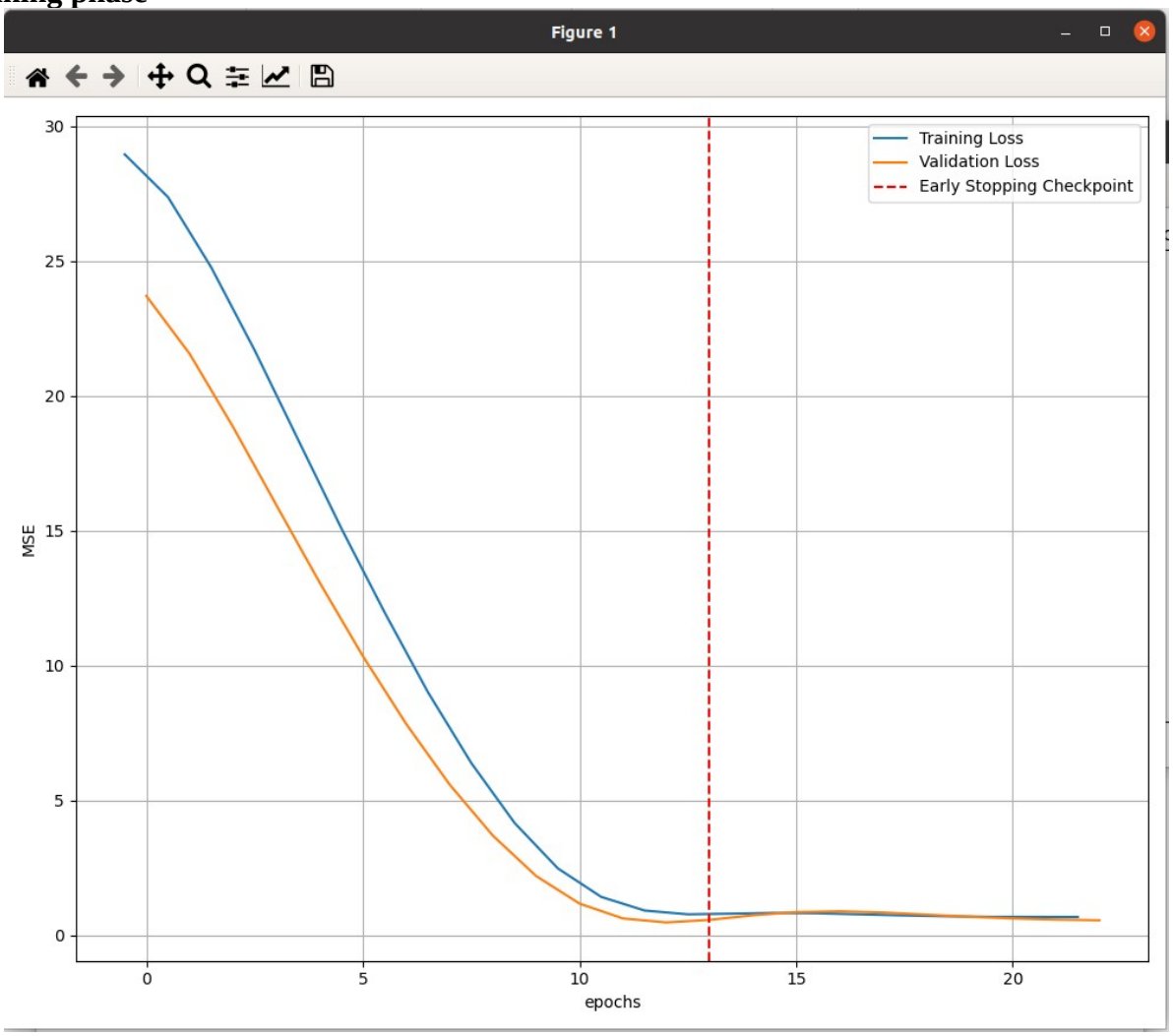
- **Regression δίκτυο:**

Υπερπαράμετροι:
#κόμβων κρυφού επιπέδου : 6 , max_epochs=100 ,batch_size = 256 , learning rate = 0.001 ,momentum = 0.9 , patiance = 10
και optimizer Stochastic gradient decsent.

Μετά το τρέξιμο του 5-fold CV πήραμε:
The average MSE obtained over all the fold validations (last epoch) is:
0.688030099

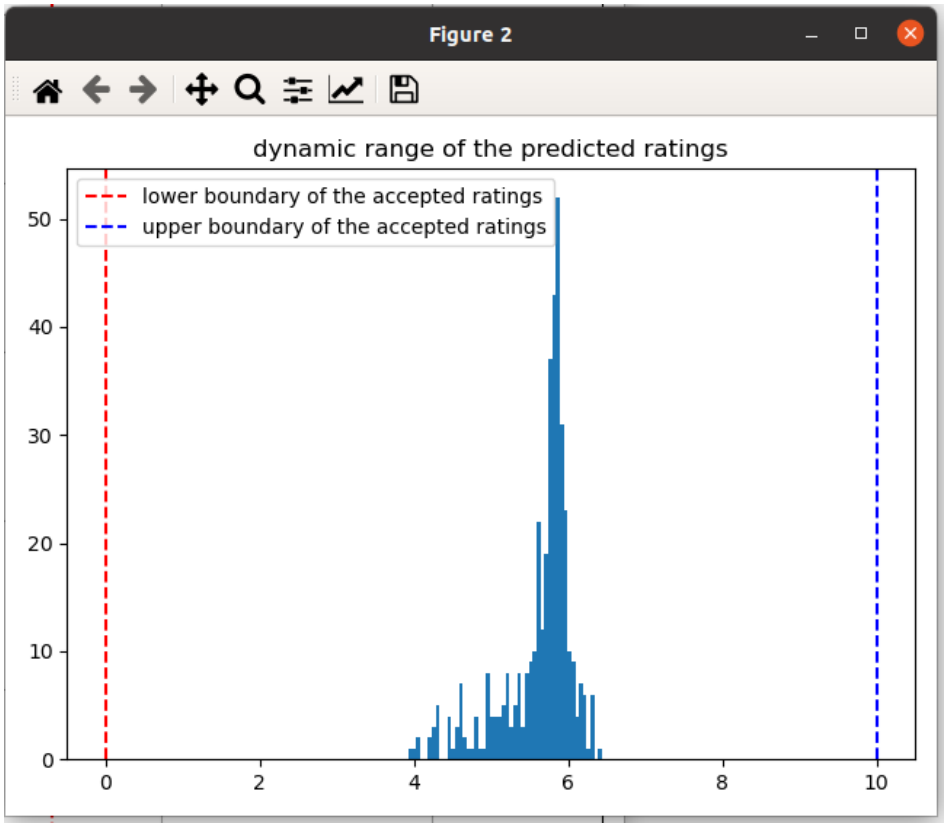
Μετά την εκπαίδευση για τις παραπάνω υπερπαραμέτρους η καμπύλη μάθησης έχει ως εξής:

training phase



Μετά από την τροφοδότηση των άγνωστων εισόδων (test set) στο δίκτυο έχουμε:

testing phase



Παρατηρούμε ότι αυτό το μικρό (σε πλήθος παραμέτρων) δίκτυο που εκπαιδεύσαμε έχει παρόμοια αποτελέσματα με το linear regression αν και θα μπορούσαμε να προσθέσουμε και άλλα επίπεδα μήπως έτσι καταφέρναμε να κάναμε καλύτερο feature extraction …
Όσον αφορά τις υπερπαραμέτρους έχουμε:

- Οι κόμβοι επιλέχθηκαν 6 έτσι ώστε να κάνουμε καποιου είδους feature extraction (παρατηρήσαμε ότι οι περισσότερες διαστάσεις δεν βοηθήσαν ιδιαίτερα).
- Το learning rate υπολογίσθηκε με εμπειρικό τρόπο αν και υπάρχει μαθηματική θεμελίωση και η οποία δίνει ένα άνω φράγμα κάτω από το οποίο ο αλγόριθμος θα συγκλίνει (βγαίνει από ιδιοανάλυση του Μητρώου αυτοσυσχέτισης των δεδομένων) . Όσο μεγαλύτερο τόσο ο αλγόριθμος θα συγκλινει γρηγορότερο σε κάποιο τοπικό ελάχιστο αλλά με κίνδυνο να αποκλίνει (καθώς ο αλγόριθμος είναι στοχαστικός δηλαδή δεν υπολογίζει ποτέ το ακριβές ελάχιστο αλλά κατά μέση τιμή) ενώ όσο μικρότερο τότε κάνουμε μικρότερα βήματα προς το τοπικό ελάχιστο (αργότερη σύγκλιση) αλλά θα σίγουρα θα αποκλίνουμε λιγότερο από το τοπικό ελάχιστο .
- Το momentum είναι και αυτό μια παράμετρος η οποία μας βοηθά στη γρηγορότερη σύγκλιση .

- **Classification δίκτυο:**

Εξετάσαμε δύο περιπτώσεις :

- Ένα κρυφό επίπεδο με relu μη γραμμικότητα

Υπερπαράμετροι:
#κόμβων κρυφού επιπέδου : 6 (και το επίπεδο εξόδου έχει 10 κόμβους όπως σημειώθηκε και στο σχήμα παραπάνω) , max_epochs=100 ,batch_size = 256 , learning rate = 0.005 , patiance = 10 και optimizer ο Adam .

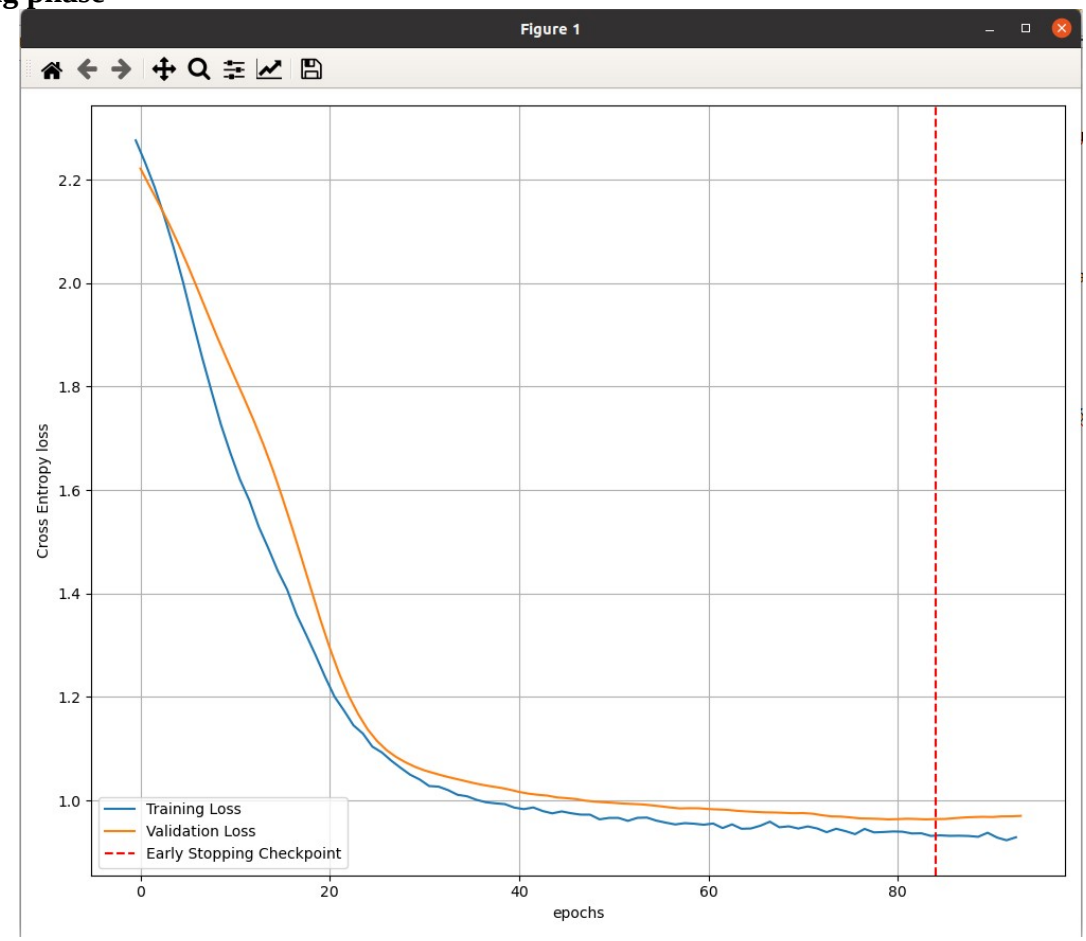
Μετά το τρέξιμο του 5-fold CV πήραμε:

The average cross entropy loss and accuracy obtained over all the fold validations (last epoch) is:

loss= 1.0031859636306764
accuracy= 0.5784483960948397

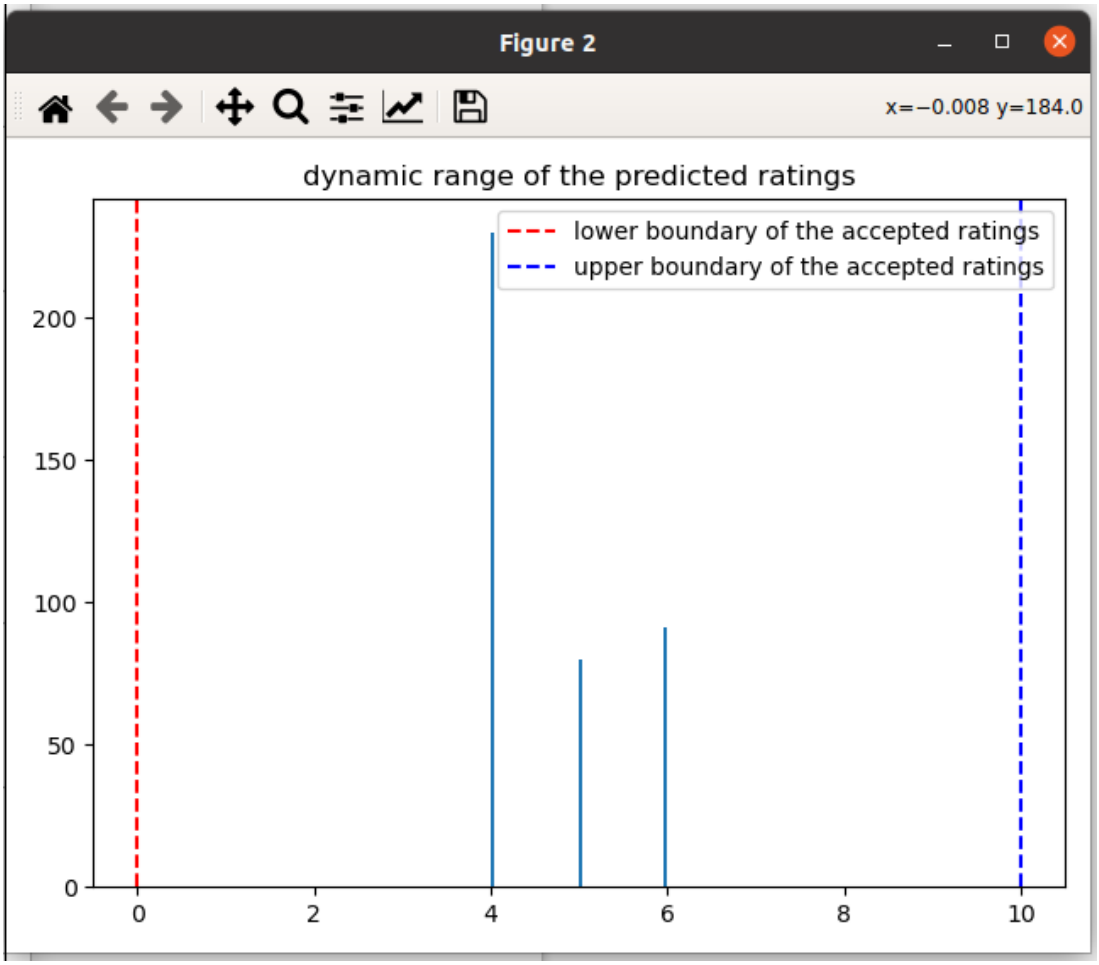
Μετά την εκπαίδευση για τις παραπάνω υπερπαραμέτρους η καμπύλη μάθησης έχει ως εξής:

training phase



Μετά από την τροφοδότηση των άγνωστων εισόδων (test set) στο δίκτυο έχουμε:

testing phase



- Κανένα κρυφό επείπεδο

Να σημειωθεί ότι αυτή η περίπτωση πρέπει να είναι ισοδύναμη με τον αλγόριθμο multinomial logistic regression καθώς οι μόνες παράμετρες που μαθαίνει το δίκτυο είναι ένα linear mapping ενώ το loss το οποίο ελαχιστοποιείται και στις δύο περιπτώσεις είναι το cross entropy.

Υπερπαράμετροι:
#κόμβων κρυφού επιπέδου : 0 (το επίπεδο εξόδου εξακολουθεί να έχει 10 κόμβους όσες και οι κλάσεις) , max_epochs=100 ,batch_size = 256 , learning rate = 0.01 , patiance = 10
και optimizer ο Adam .

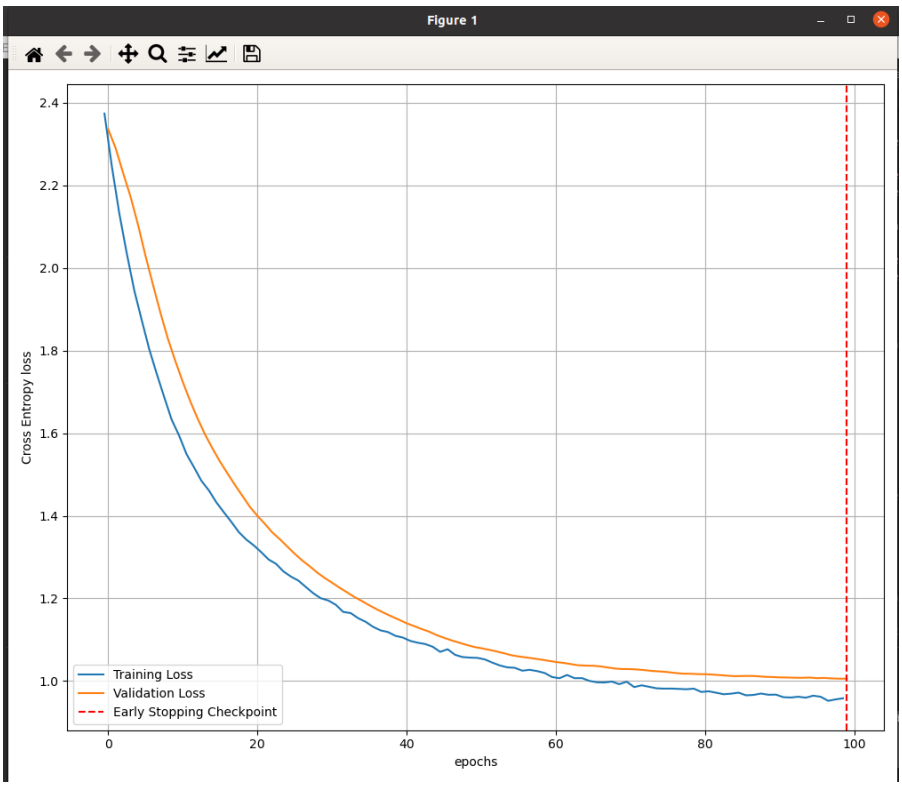
Μετά το τρέξιμο του 5-fold CV πήραμε:

The average cross entropy loss and accuracy obtained over all the fold validations (last epoch) is:

loss= 1.0269763112068175
accuracy= 0.582639470013947

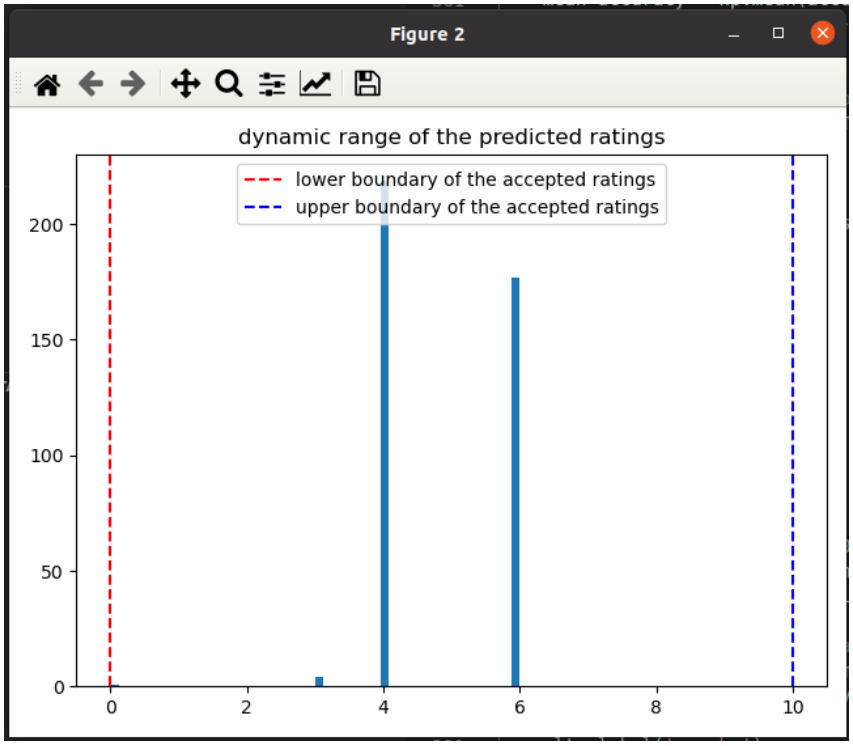
Μετά την εκπαίδευση για τις παραπάνω υπερπαραμέτρους η καμπύλη μάθησης έχει ως εξής:

training phase



Μετά από την τροφοδότηση των άγνωστων εισόδων (test set) στο δίκτυο έχουμε:

testing phase



Παρατηρούμε ότι η προσθήκη του hidden layer δεν έπαιξε σημαντικό ρόλο στη μοντελοποίηση του προβλήματος και αυτό φαίνεται τόσο από τις μετρικές στο validation set όσο και από τα αποτελέσματα στις άγνωστες εισόδους καθώς οι μεν μετρικές δεν άλλαξαν πολύ (αν και αυτό δεν μας λέει πολλά) και οι δε έξοδοι των άγνωστων εισόδων είναι συνεχίζουν να είναι biased από τις

πιο συχνά εμφανιζόμενες αξιολογίσεις του training set (αδυναμία των μοντέλων να γενικεύουν (αν και αυτό ενδέχεται να συμβαίνει κυρίως λόγο της έλλειψης δεδομένων)) . Συγκριτικά τώρα με τις απλές πιο απλές μεθόδους classification που εξετάσαμε στα πλαίσια τις εργασίας είδαμε ότι η επιλογή ενός πιο πολύπλοκου αλγορίθμου δεν σημαίνει απαραίτητα και ότι θα πάρουμε καλύτερες προβλέψεις . Επομένως καταλήγουμε στο ότι η επιλογή του αλγορίθμου που θα χρησιμοποιήσουμε για να λύσουμε ένα πρόβλημα θα εξαρτάται πολύ από το πρόβλημα αλλά και την ποσότητα και την ποιότητα των δεδομένων μας .