ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής

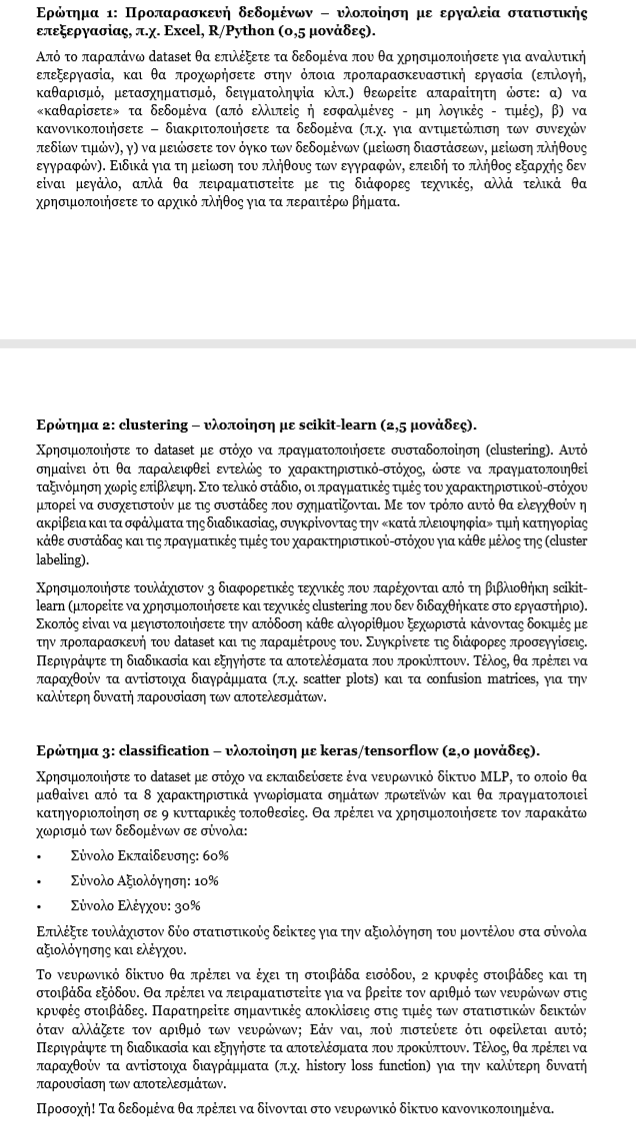


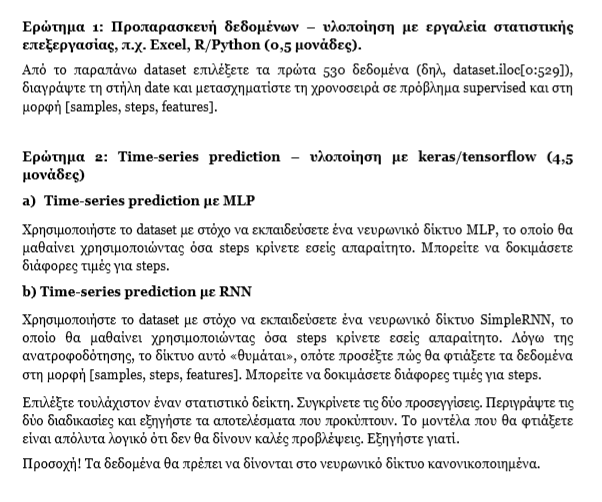
Εργασία Μαθήματος «ΑΝΑΛΥΤΙΚΗ ΔΕΔΟΜΕΝΩΝ»

|  |  |
| --- | --- |
| 2019-2020 | Εργασία Αναλυτική Δεδομένων |
| Όνομα φοιτητή – Αρ. Μητρώου  (όλων σε περίπτωση ομαδικής εργασίας) | ΣΤΕΦΑΝΟΣ ΠΑΣΙΟΣ – Π17102 |
| ΔΗΜΗΤΡΗΣ ΠΑΠΑΔΗΜΗΤΡΙΟΥ – Π17097 |
| Ημερομηνία παράδοσης | 08/07/2020 |

**Εκφώνηση της άσκησης**

Στην ενότητα αυτή περιλαμβάνεται η εκφώνηση της άσκησης.

****

****

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

[1 ΜΕΡΟΣ A 4](file:///C:\Users\stefa\Downloads\Template-JAVA%20NET%20PROG.doc#_Toc308529529)

[1.1 Προπαρασκευή δεδομένων – υλοποίηση με εργαλεία στατιστικής επεξεργασίας, π.χ. Excel, R/Python 4](file:///C:\Users\stefa\Downloads\Template-JAVA%20NET%20PROG.doc#_Toc308529530)

1.2 clustering – υλοποίηση με scikit-learn 15

1.3 classification – υλοποίηση με keras/tensorflow 35

[2 ΜΕΡΟΣ Β…………………………………………………………………………………………..……………………….51](file:///C:\Users\stefa\Downloads\Template-JAVA%20NET%20PROG.doc#_Toc308529532)

[2.1 Προπαρασκευή δεδομένων – υλοποίηση με εργαλεία στατιστικής επεξεργασίας, π.χ. Excel, R/Python 4](file:///C:\Users\stefa\Downloads\Template-JAVA%20NET%20PROG.doc#_Toc308529530)

2.2 Time-series prediction – υλοποίηση με keras/tensorflow 9

**ΑΝΑΛΥΤΙΚΗ ΔΕΔΟΜΕΝΩΝ**

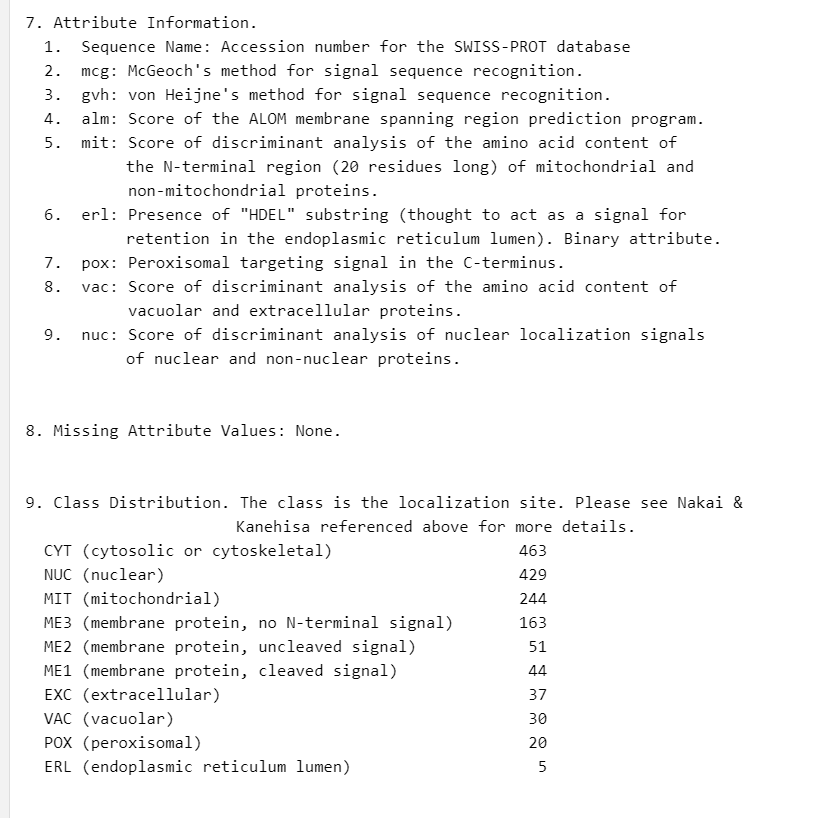
**Απαλλακτική Εργασία**

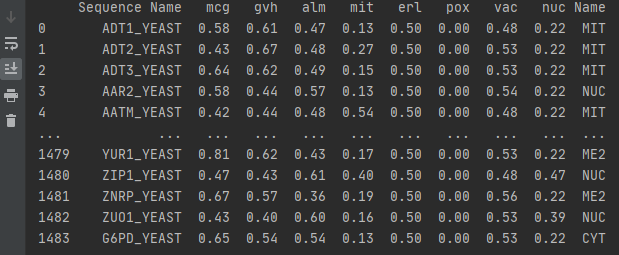
**(Ομάδες των 1-3 ατόμων)**

**1. ΜΕΡΟΣ Α**

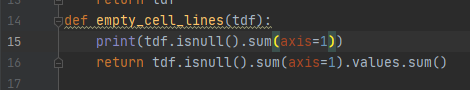
**1.1 Προπαρασκευή δεδομένων – υλοποίηση με εργαλεία στατιστικής επεξεργασίας, π.χ. Excel, R/Python.**

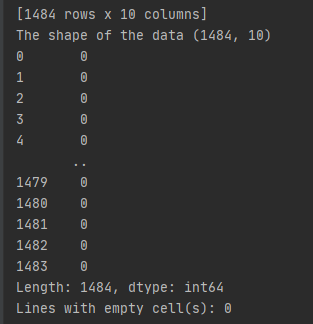
Αρχικά , από το yeast.names μπορούμε να αντλήσουμε της εξής πληροφορία για το dataset.



Υπάρχουν δηλαδή 9 πεδία + 1 τα labels. Φορτώνοντας το dataset με python (με τη βοήθεια της βιβλιοθήκης Pandas) έχουμε αποτελέσματα της εξής μορφής:

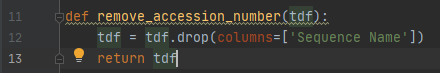
Στο αρχείο .names με τις πληροφορίες στον αριθμό 8. δίνει την πληροφορία πως δεν υπάρχει κανένα κενό πεδίο στο dataset ωστόσο μπορούμε να το ελέγξουμε και μόνοι



****

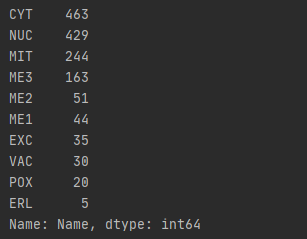
Ότι όντως ισχύει άρα δεν χρειάζεται κάποια ενέργεια.

Στη συνέχεια το πρώτο κελί το οποίο είναι unique identifier που χρησιμεύει για την βάση δεδομένων το αφαιρούμε τελείως μιας και δεν θα έχει κάποια χρήση για την ανάλυση έτσι ώστε να μειωθούν και οι διαστάσεις.

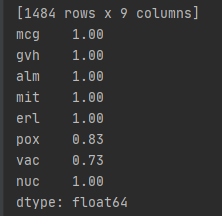


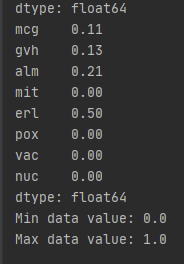


Επιπλέον ελέγχονται και τα ονόματα τα οποία είναι όντως σύμφωνα με το .names οπότε δεν υπάρχει κάτι εσφαλμένο.



Ενώ για τα υπόλοιπα δεδομένα υπολογίζεται το min και max value



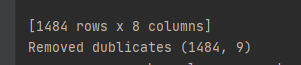
****

Όπου όπως φαίνεται είναι στο πεδίο 0-1 για όλες τις στήλες χαρακτηριστικών 'mcg','gvh','alm','mit','erl','pox','vac','nuc'.

Με τη παρακάτω συνάρτηση

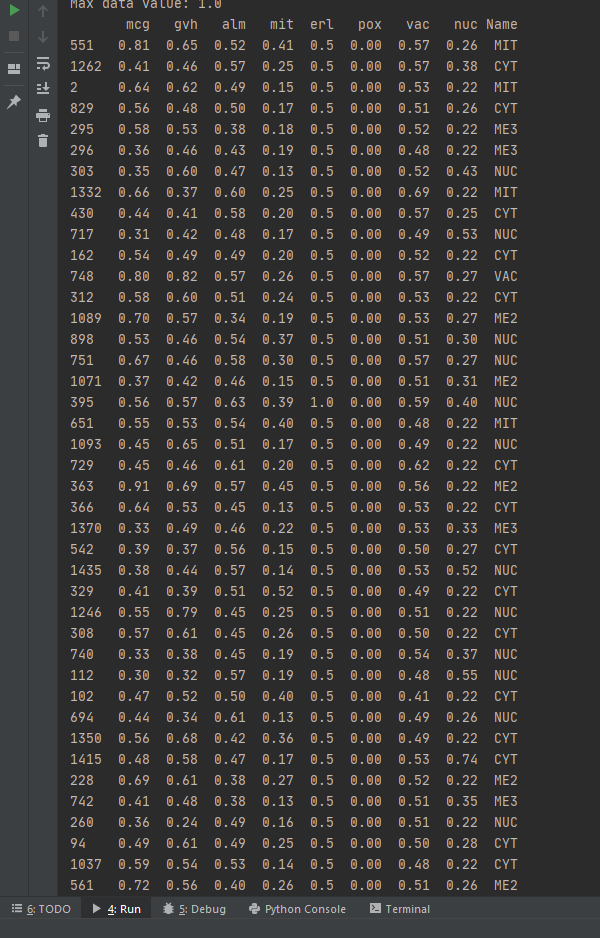


αφαιρούνται οι διπλές εγγραφές και μένει μόνο η πρώτη από τις διπλές εγγραφές.



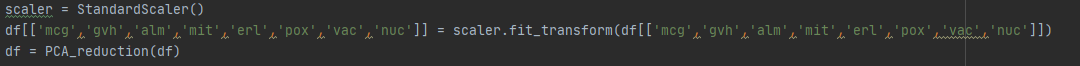
Ωστόσο και πάλι στα δεδομένα δεν υπάρχει κάποια εγγραφή που να έχει διπλές γιατί όπως φαίνεται οι διαστάσεις παραμένουν οι ίδιες.

Για την μείωση των εγγραφών μπορεί να χρησιμοποιηθεί η τεχνική του random sampling έστω για το 25% του συνόλου των εγγραφών των δεδομένων αλλά επειδή έτσι και αλλιώς οι εγγραφές είναι λίγες (1483 συνολικά) στο τέλος δεν κρατιέται το αποτέλεσμα απλά τυπώνεται.

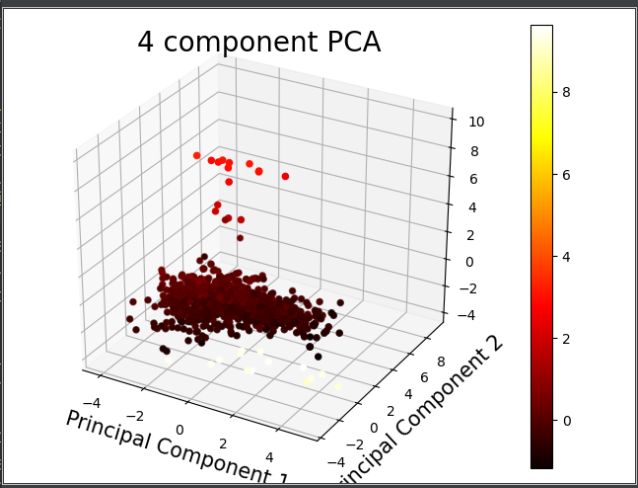


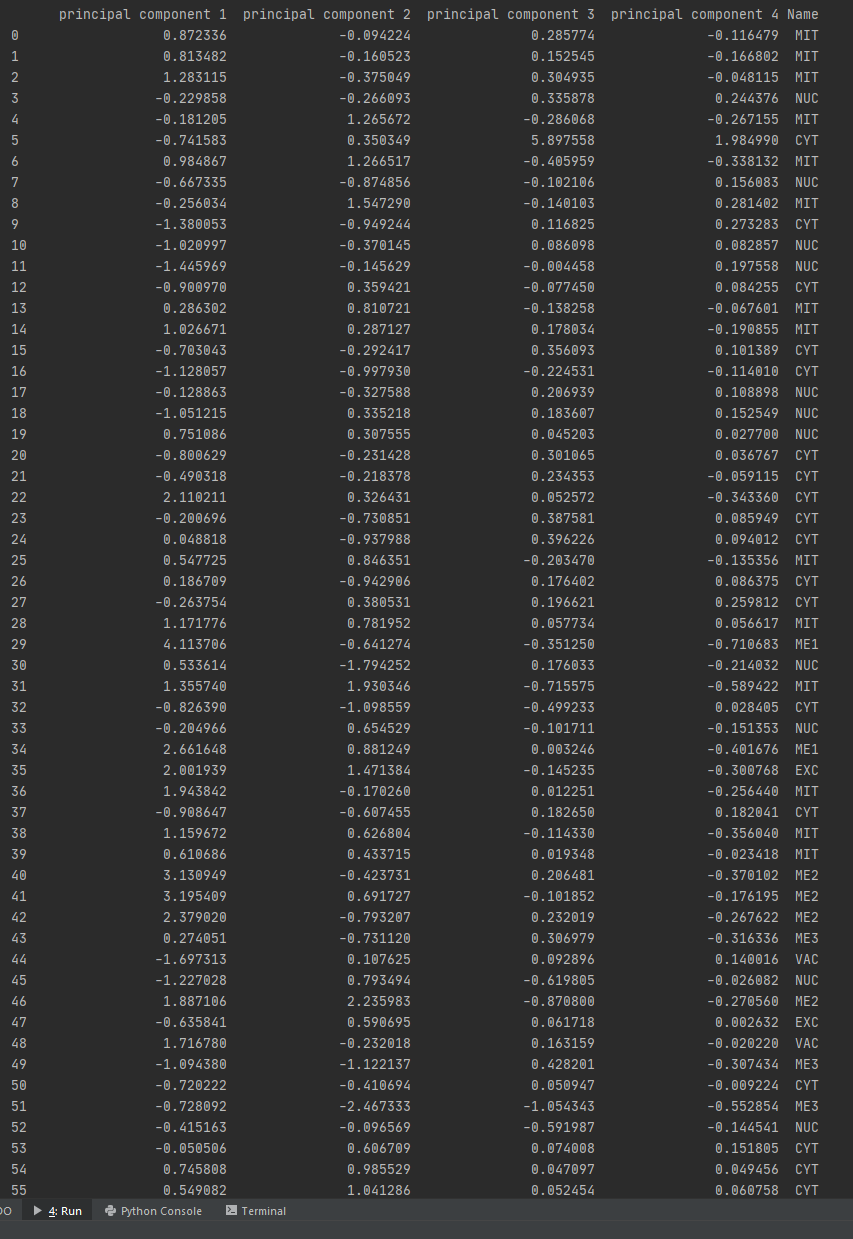
Για να μειωθεί ο αριθμός των πεδίων θα χρησιμοποιηθεί η τεχνική PCA (principal component analysis) και από 8 που είναι αρχικά θα τα μειώσουμε σε 4 μιας και εκεί φαίνεται να δίνουν τα βέλτιστα αποτελέσματα.

Το PCA επηρεάζεται κατά κλίμακα, οπότε πρέπει να κλιμακώσουμε τις δυνατότητες στα δεδομένα μας πριν από την εφαρμογή PCA. Χρησιμοποιώντας το StandardScaler για να τυποποιήσουμε τις δυνατότητες του συνόλου δεδομένων σε κλίμακα μονάδας (μέσος όρος = 0 και διακύμανση = 1) που αποτελεί απαίτηση για τη βέλτιστη απόδοση πολλών αλγορίθμων μηχανικής μάθησης.

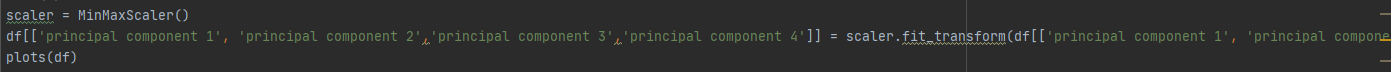


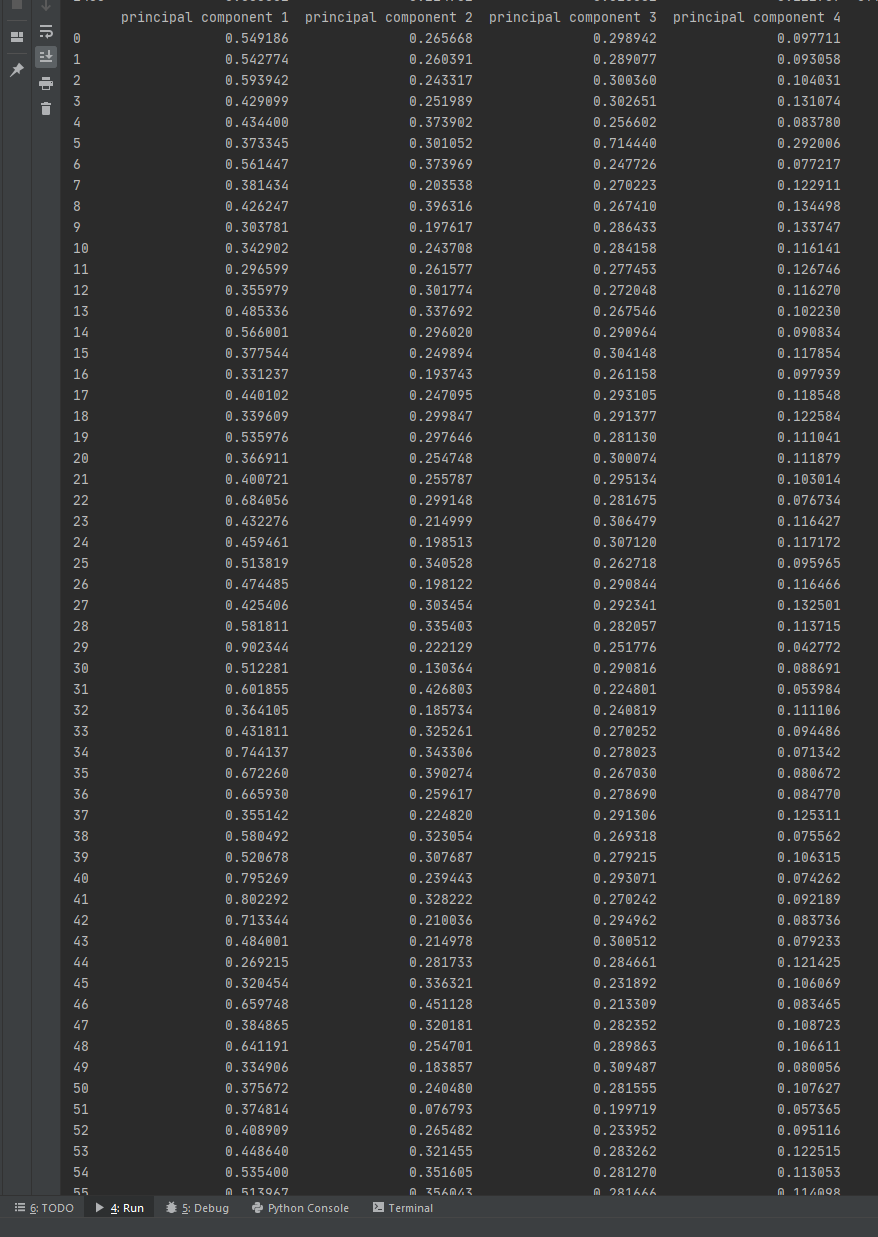




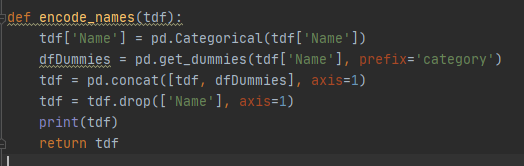


Επειδή , πλέον υπάρχουν τιμές που ξεφεύγουν από το πεδίο 0-1 ή -1-1 όπως +4 , -2 κτλπ θα γίνει κανονικοποίηση με τον MinMaxScaler στο πεδίο 0-1



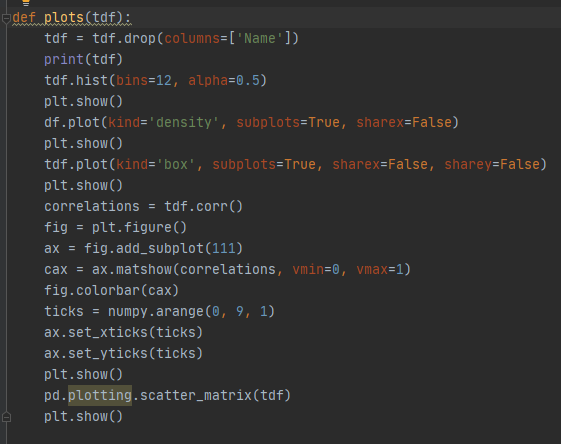


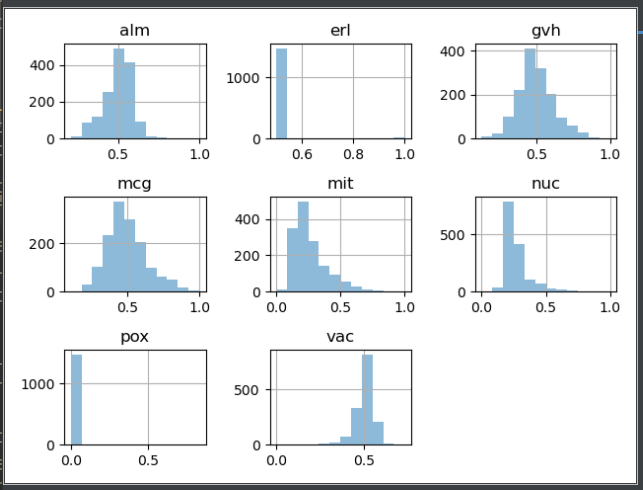
Επειδή ο υπολογιστής δεν μπορεί να καταλάβει το ονόματα CYT,NUC,MIT,ME3,ME2,ME1,EXC,VAC,POX,ERL πρέπει να γίνει αριθμητική αναπαράσταση. Υπάρχουν αρκετοί τρόποι για να γίνει αυτό αλλά ένας από τους καλύτερους είναι ο One Hot Encoder. Δηλαδή το CYT θα είναι [1,0,0,0,0,0,0,0,0,0] το NUC [0,1,0,0,0,0,0,0,0,0] κτλπ.

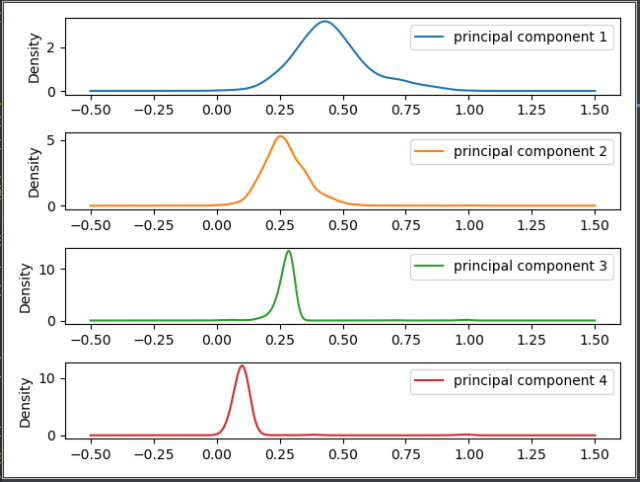


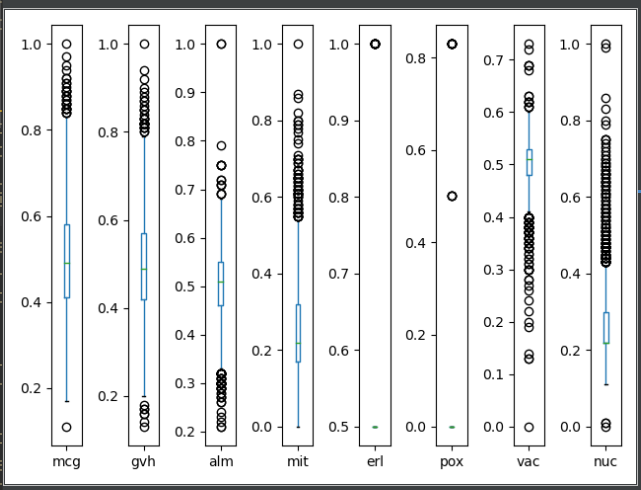
****

Επιπλέον εμφανίζονται διαγράμματα τόσο στην αρχή όσο και το τέλος της επεξεργασίες των δεδομένων όπως histogram με 12 bins , destiny plot , box plot κτλπ.

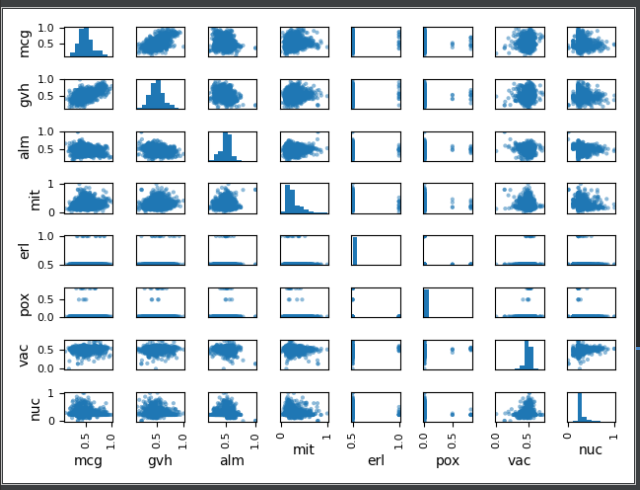




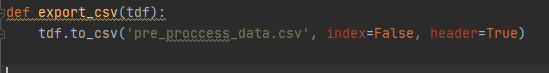


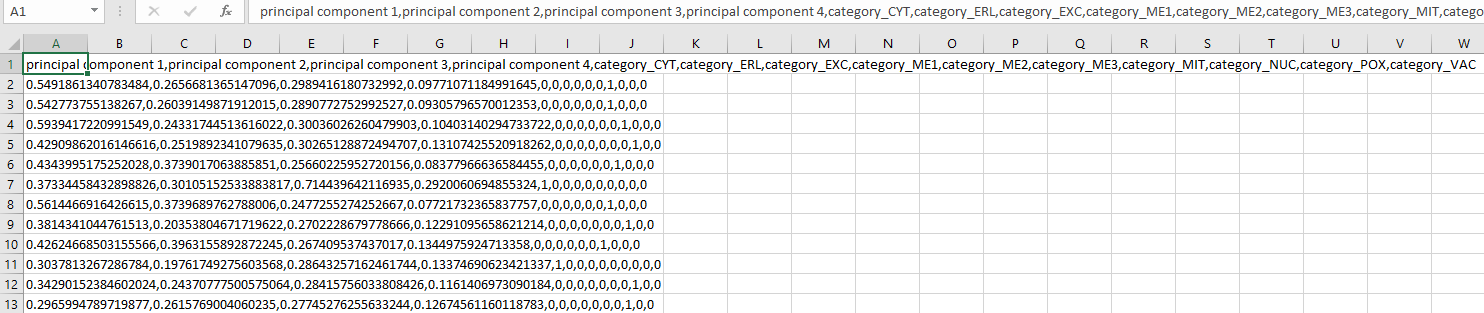






Το πρόγραμμα έχει ως output ένα νέο αρχείο CSV με τα νέα δεδομένα

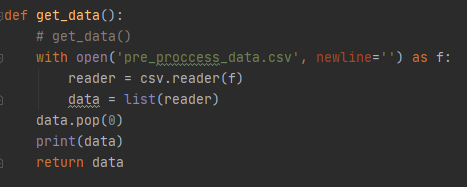




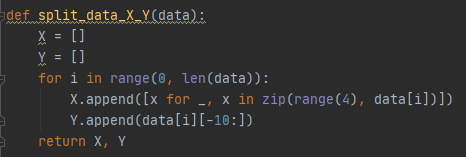
**1.2 clustering – υλοποίηση με scikit-learn.**

Το αρχείο clustering.py υλοποιεί το ερώτημα 2.

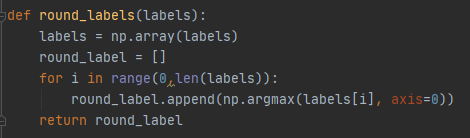
Αρχικά με την συνάρτηση



διαβάζεται το CSV αρχείο που έχει προκύψει από το ερώτημα 1 με τα δεδομένα και αποθηκεύονται σε μια λίστα. Στη συνέχεια με τη συνάρτηση

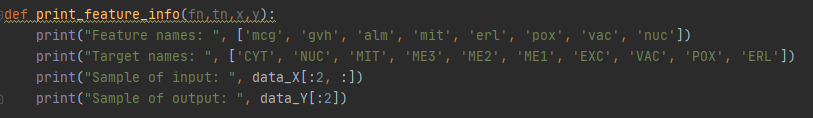


σπάνε και διαχωρίζονται τα δεδομένα και τα labels σε hot encoded μορφή σε δύο διαφορετικές λίστες X,Y. Επειδή , οι αλγόριθμοι clustering έχουν ως output labels της μορφής 0,1,2,3,… με τη συνάρτηση



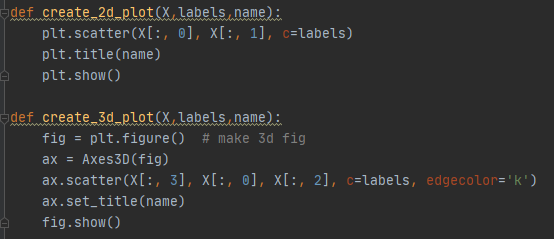
τη λίστα Y από hot encoded μορφή σε παρόμοια με αυτή των αλγορίθμων. Συγκεκριμένα το [1,0,0,0,0,0,0,0,0,0] θα είναι ίσο με 0 το [0,1,0,0,0,0,0,0,0,0] ίσο με 1 και τα λοιπά. Δηλαδή ανάλογα με την θέση που βρίσκεται ο άσσος.

H συνάρτηση



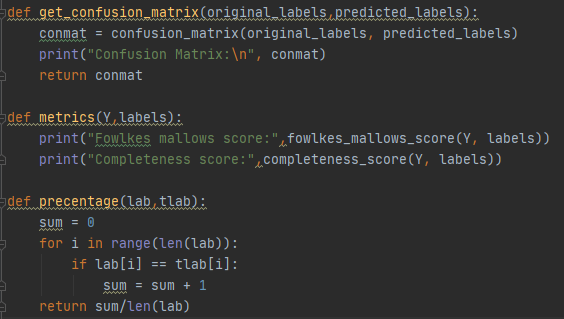
τυπώνει ως παράδειγμα πως είναι τα input και τα output

ενώ οι συναρτήσεις



δέχονται τα διανύσματα X , τα labels και το όνομα που θέλουμε να ορίσουμε για το plot και δημιουργούν 3d και 2d plots.

Με τις παρακάτω 3 συναρτήσεις μετράμε τις αποδόσεις των αλγορίθμων



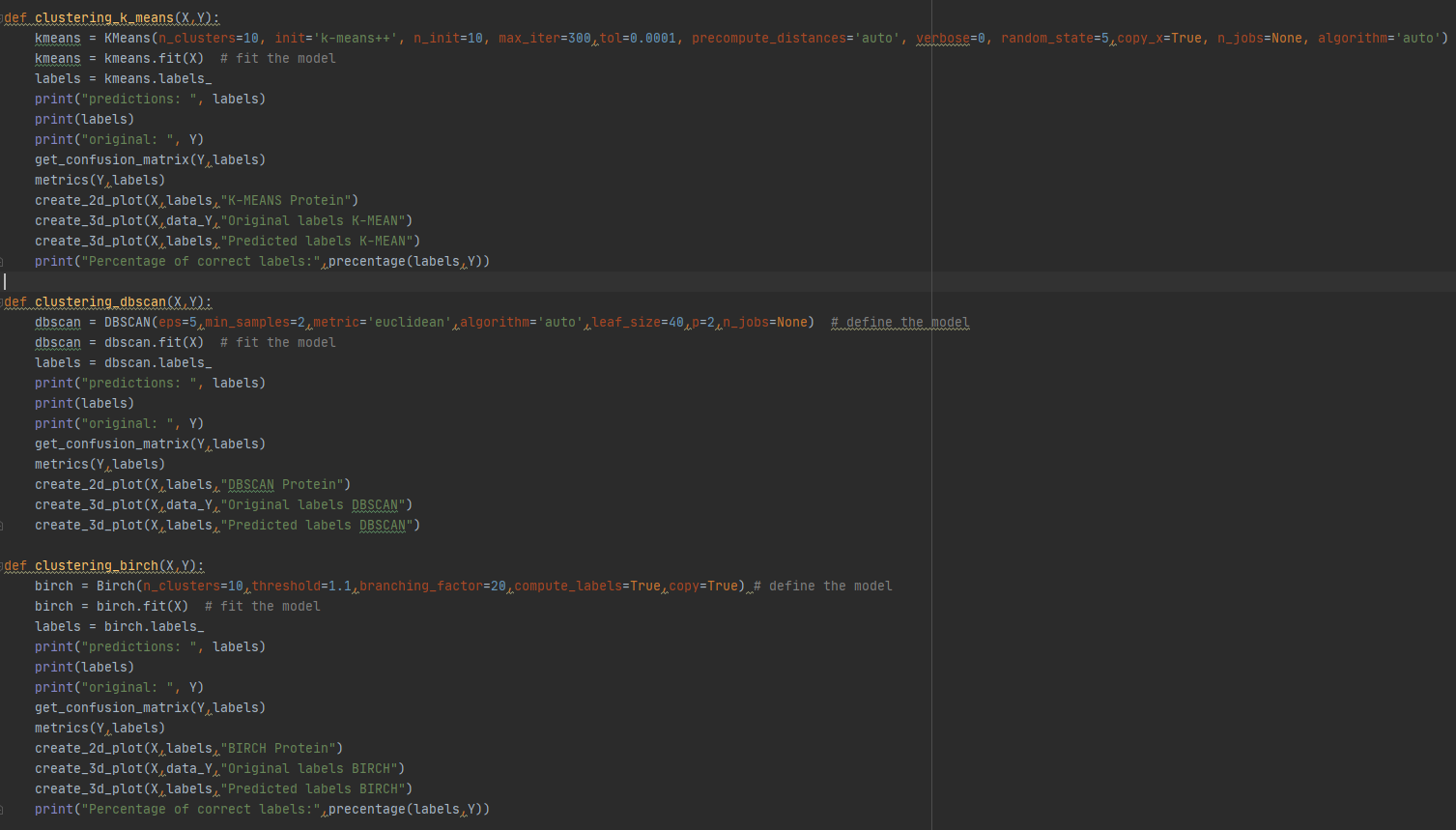
**Η πρώτη** από αυτές επιστρέφει τον confusion matrix. Η πληροφορία που μας δίνει ο confusion matrix είναι ότι οι γραμμές είναι το Predicted value και οι στήλες το actual value. Δηλαδή στη θέση [0,0] μας λέει ότι η τιμή που προβλέφθηκε από το clustering είναι 0 και η αληθινή τιμή είναι 0. Στο [0,1] ότι η τιμή που προβλέφθηκε από το clustering είναι 0 και η αληθινή τιμή είναι 1. Δηλαδή στη κύρια διαγώνιο είναι όλα αυτά που προβλέφθηκαν σωστά ενώ σε όλα τα υπόλοιπα κελιά τα λανθασμένα.

**Η δεύτερη** χρησιμοποιεί κάποιες από τις μετρικές που παρέχει η βιβλιοθήκη scikit-learn για clustering οι οποίες μπορούν να βρεθούν τον παρακάτω σύνδεσμο

<https://scikit-learn.org/stable/modules/model_evaluation.html>

**Η τρίτη** μετράει το ποσοστό τον επιτυχημένων προβλέψεων το οποίο υπολογίζεται ως ο αριθμός των σωστών δια τον συνολικό αριθμό δεδομένων.

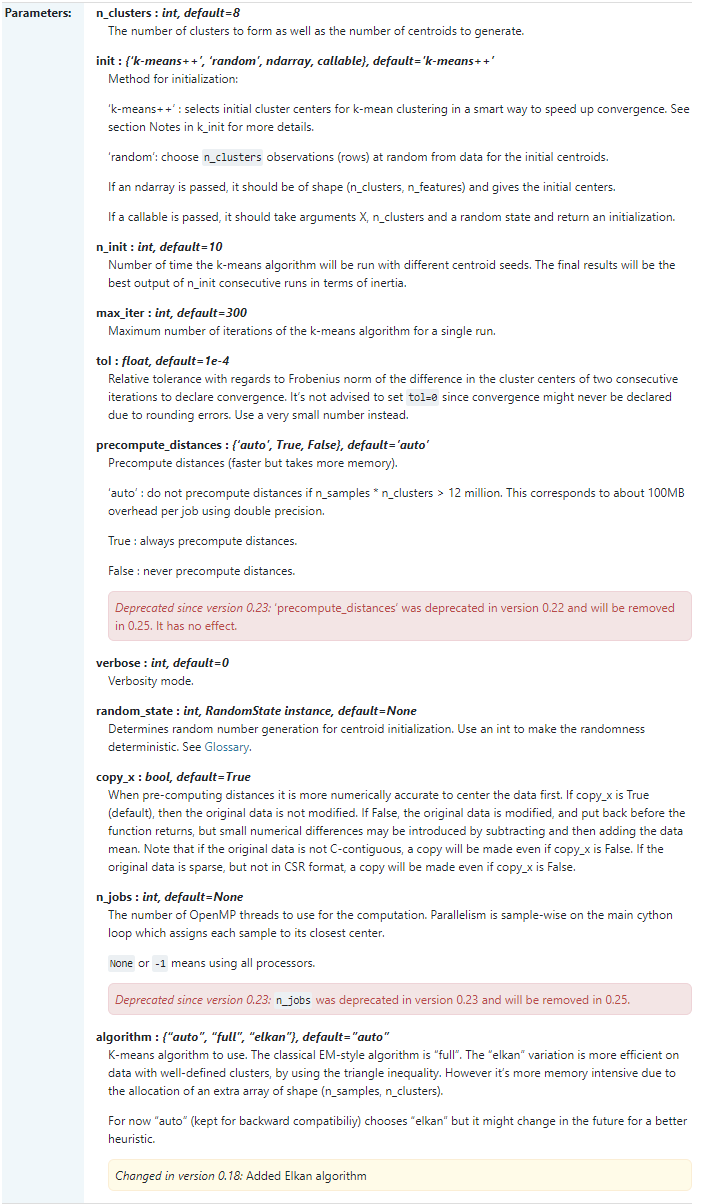
Τέλος οι συναρτήσεις

****

Υλοποιούν τους 3 αλγορίθμους clustering k\_means,dbscan,birch με τις ανάλογες μεθόδους που παρέχει η βιβλιοθήκη scikit-learn ενώ χρησιμοποιούν και τις μεθόδους που αναφέρθηκαν παραπάνω για να εμφανισθούν τα plots , τα confusion matrix αλλά και η απόδοση των αλγορίθμων.

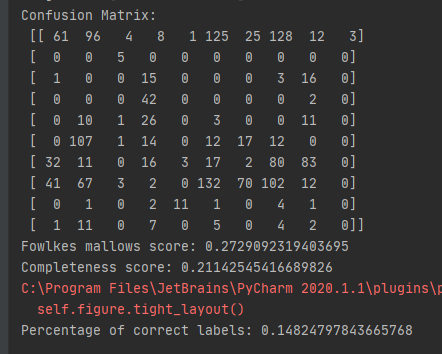
Ο πρώτος αλγόριθμος που υλοποιήθηκε είναι ο k-means. Το k-means clustering είναι μια μέθοδος κβαντισμού διανύσματος, αρχικά από την επεξεργασία σήματος, που στοχεύει να χωρίσει n παρατηρήσεις σε k clusters όπου κάθε παρατήρηση ανήκει στο σύμπλεγμα με τον πλησιέστερο μέσο όρο (cluster centres ή cluster centroid), που χρησιμεύει ως πρωτότυπο του το σύμπλεγμα. Αυτό έχει ως αποτέλεσμα την κατάτμηση του χώρου δεδομένων σε κελιά Voronoi.Το k-means clustering ελαχιστοποιεί τις διακυμάνσεις εντός του συμπλέγματος (αποστάσεις Euclidean square), αλλά όχι τις κανονικές αποστάσεις Euclidean, που θα ήταν το πιο δύσκολο πρόβλημα Weber: ο μέσος όρος βελτιστοποιεί τα τετράγωνα σφάλματα, ενώ μόνο η γεωμετρική διάμεση ελαχιστοποιεί τις αποστάσεις Euclidean. Για παράδειγμα, μπορούν να βρεθούν καλύτερες λύσεις Ευκλείδειας χρησιμοποιώντας k-medians και k-medoids.

Το σύνολο των ορισμάτων (αλλά και ο ρόλος τους) που δέχεται σύμφωνα το scikitlearn είναι:



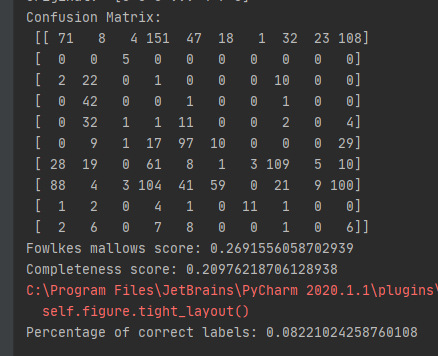
kmeans = KMeans(n\_clusters=10, init='k-means++', n\_init=10, max\_iter=300,tol=0.0001, precompute\_distances='auto', verbose=0, random\_state=None,copy\_x=True, n\_jobs=None, algorithm='auto')

Αρχικά εκτελούμε τον αλγόριθμο με τις default τιμές εκτός από το n\_clusters που το ορίζουμε 10 αφού έχουμε συνολικά 10 διαφορετικά labels CYT,NUC,MIT,ME3,ME2,ME1,EXC,VAC,POX,ERL (όπως στο εργαστήριο για τα iris δεδομένα που είχαν 3 κλάσεις ορίζαμε k=3)

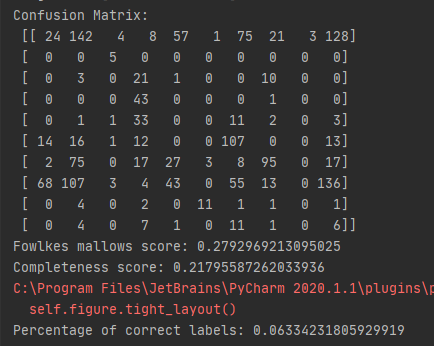


Χρησιμοποιούμε κάποιες μετρικές που παρέχει το scikit-learn για clustering αλλά υπολογίζουμε και το ποστοστό μόνοι ως ο αριθμός των σωστών / τον συνολικό αριθμό.

Αρχικά δοκιμάζουμε να αλλάξουμε το init από την default τιμή (k-means++) σε random αλλά το ποσοστό πέφτει οπότε και επιστρέφουμε στο αρχικό

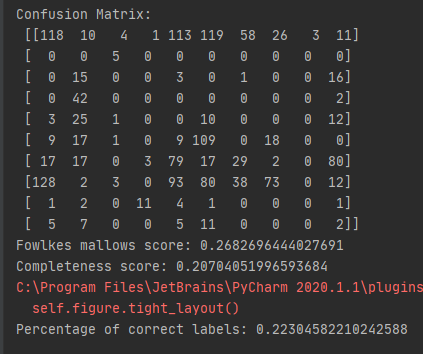


Στη συνέχεια αλλάζουμε το random\_state από None σε 2.



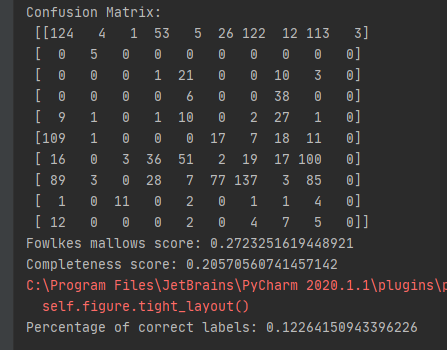
Παρατηρείται ότι το ποσοστό είναι χειρότερο από πριν.

Για random\_state = 5



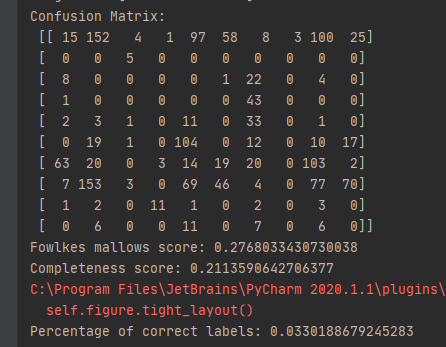
Το ποσοστό βελτιώνεται.

Για random\_state=6 αλλά και για τις υπόλοιπες μεγαλύτερες τιμές το ποσοστό μειώνεται

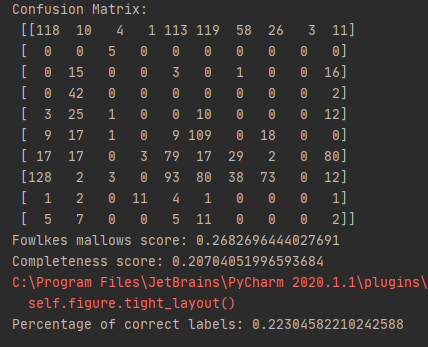


Οπότε κρατάμε random\_state = 5.

Για το n\_init αν το μειώσουμε πχ n\_init = 5



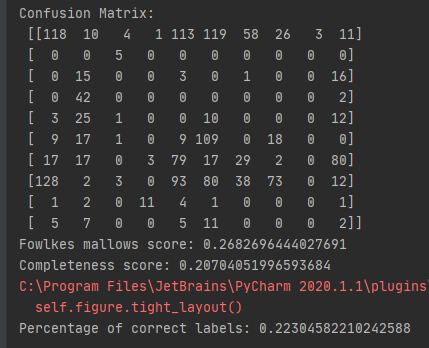
Η απόδοση πέφτει ενώ για μεγαλύτερα n\_init (από 10-22) πχ n\_init = 20



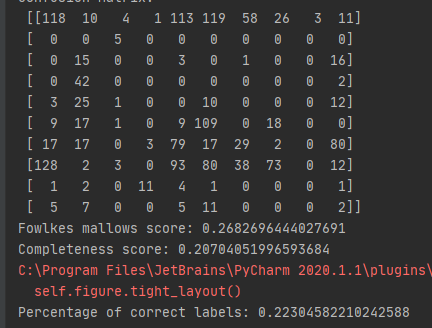
Η απόδοση παραμένει το ίδιο ενώ για >22 η απόδοση πέφτει οπότε κρατάμε τη default τιμή 10.

Για το max\_iter από τι φαίνεται είτε το αυξήσουμε είτε το μειώσουμε παραμένει το ίδιο

Max\_iter = 100



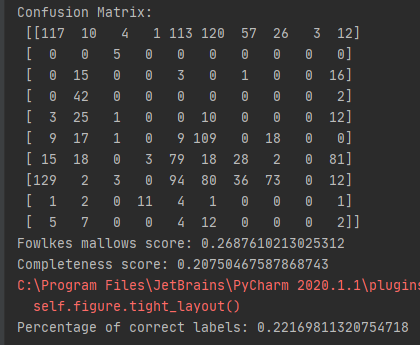
Max\_iter = 600



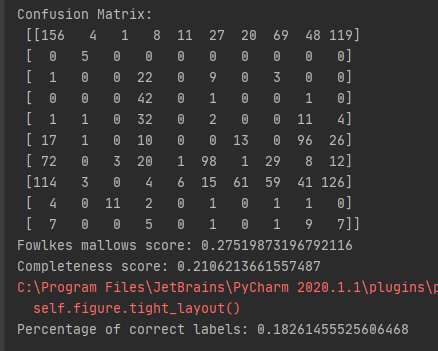
Οπότε το κρατάμε και αυτό ως έχει με την default τιμή του που είναι 300.

Η παράμετρος tol επίσης αν πειραχτεί ρίχνει την απόδοση

tol = 0.0010

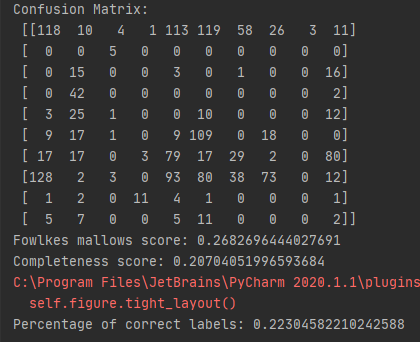


Tol = 0.1



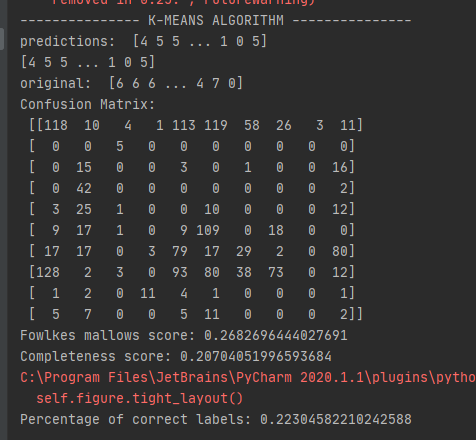
Οπότε το κρατάμε και αυτό default 0.0001.

Για τις υπόλοιπες παραμέτρους και να γίνει κάποια αλλαγή το αποτέλεσμα παραμένει πάντα το ίδιο έστω γίνονται οι εξής αλλαγές verbose = 1 , precompute\_distances = ‘***True’ ,*** copy\_x = false , n\_jobs = 16 , algorithm = ***“elkan”***

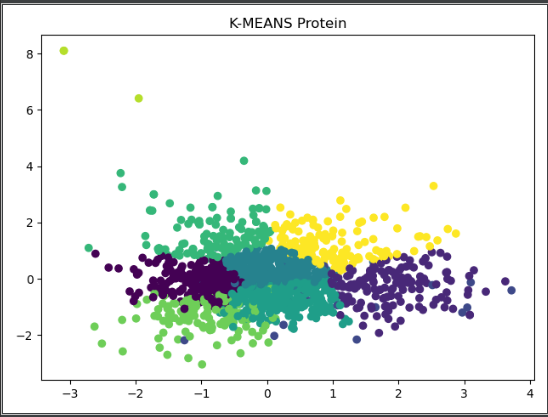
******

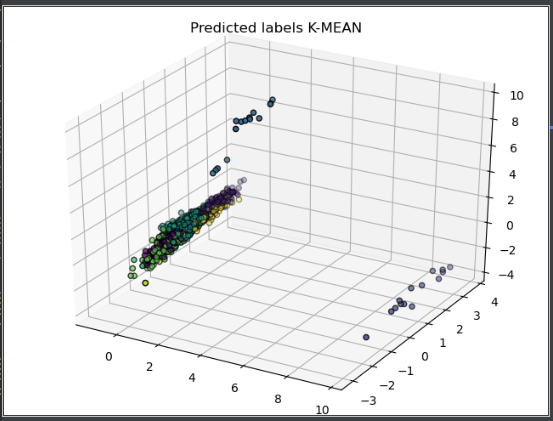
Οπότε τα αφήνουμε όλα default.

Τα τελικά αποτελέσματα



Όπου στο confusion matrix αν πάρουμε την κύρια διαγώνιο του πίνακα είναι 118+109+29+73+2 = 331 σωστές προβλέψεις ενώ σε όλα τα υπόλοιπα κελιά είναι ο λανθασμένες.





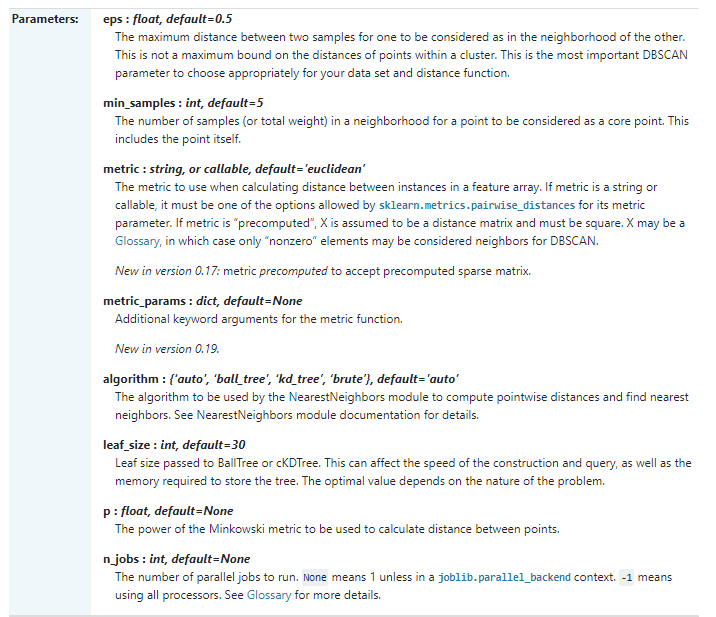
Τα παραπάνω διαγράμματα δείχνουν τις συστάδες που σχηματίζονται με τον αλγόριθμο με διαφορετικό χρώμα για κάθε συστάδα τόσο σε δυσδιάστατο όσο σε τρισδιάστατο διάγραμμα.

Ο επόμενος αλγόριθμος είναι ο **dbscan**.

Το σύνολο των ορισμάτων (αλλά και ο ρόλος τους) που δέχεται σύμφωνα το scikitlearn είναι:

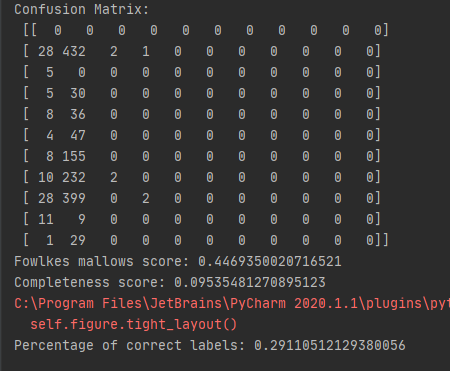
O dbscan είναι ένας μη παραμετρικός αλγόριθμος ομαδοποίησης που βασίζεται στην πυκνότητα: δεδομένου ενός συνόλου σημείων σε κάποιο χώρο, ομαδοποιεί σημεία που είναι γεμάτα μεταξύ τους (σημεία με πολλούς γειτονικούς γείτονες), επισημαίνοντας ως σημεία ακραίων τιμών που βρίσκονται μόνα τους σε περιοχές χαμηλής πυκνότητας (των πλησιέστερων γειτόνων είναι πολύ μακριά). Ο DBSCAN είναι ένας από τους πιο κοινούς αλγόριθμους ομαδοποίησης.

Ο DBSCAN δεν απαιτεί από κάποιον να καθορίσει τον αριθμό συστάδων (σε αντίθεση με τον k-means και birch) στα δεδομένα εκ των προτέρων, σε αντίθεση με το k-means. Το DBSCAN μπορεί να βρει αυθαίρετα σχήματα συστάδων. Μπορεί ακόμη και να βρει ένα σύμπλεγμα που περιβάλλεται πλήρως από (αλλά δεν είναι συνδεδεμένο) από ένα διαφορετικό σύμπλεγμα.

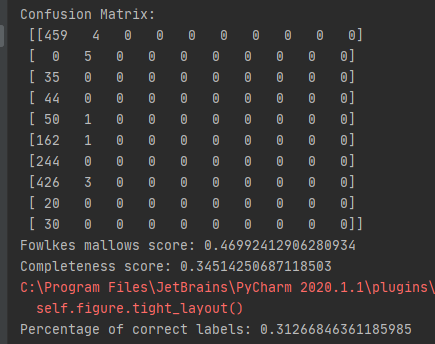


dbscan = DBSCAN(eps=0.5,min\_samples=5,metric='euclidean',algorithm='auto',leaf\_size=30,p=None,n\_jobs=None)

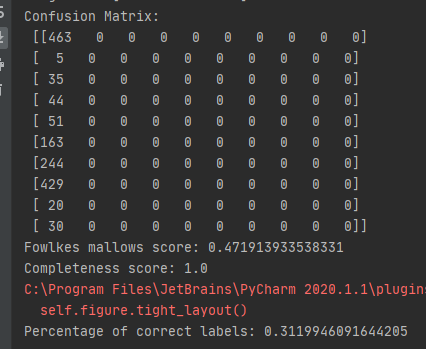
αρχικά τον τρέχουμε με τις default τιμές.



Για το όρισμα eps αν το αλλάξουμε από 5-9 μας δίνει καλύτερη απόδοση και όλες οι τιμές από 5-9 δίνουν το ίδιο αποτέλεσμα



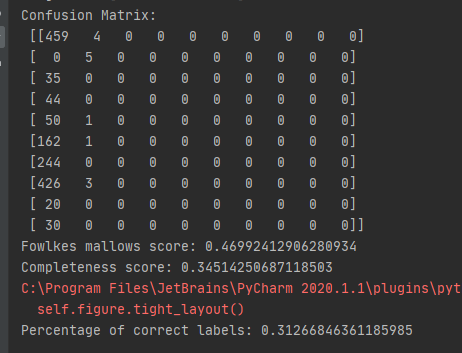
Ενώ από 10 και μετά η απόδοση αρχίζει πάλι να πέφτει



Οπότε κρατάμε esp = 5

Τα ορίσματα algorithm , n\_jobs και metric όπως είδαμε και στο προηγούμενο αλγόριθμο δεν επηρεάζουν καθόλου την απόδοση ενώ όμοια με διάφορες δοκιμές μετα leaf\_size , p , min\_samples δεν αλλάζει καθόλου η απόδοση του αλγόριθμου

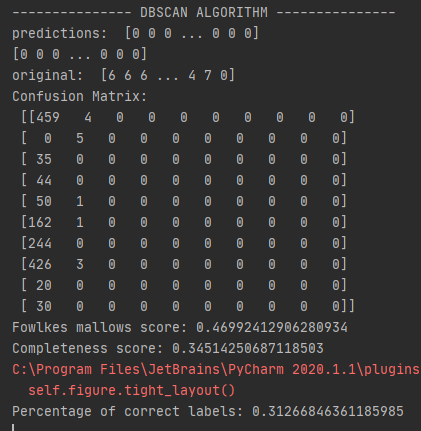
Έστω για παράδειγμα min\_samples=2 , leaf\_size = 40 , p = 2



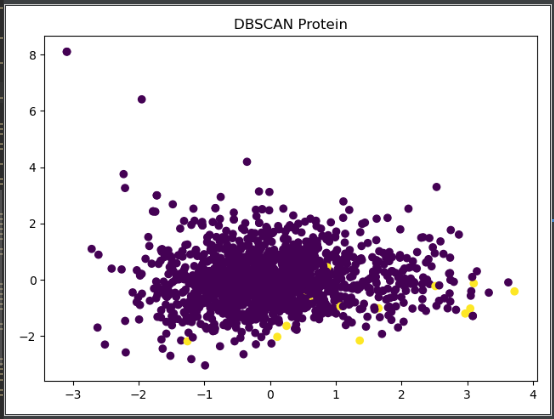
Το αποτέλεσμα παραμένει το ίδιο.

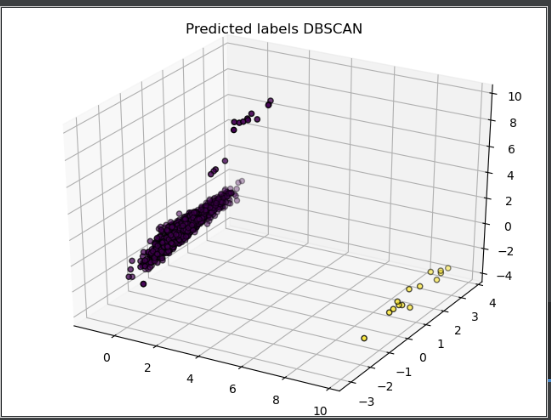
Οπότε κρατάμε όλα τα υπόλοιπα με τις default τιμές τους.

Το τελικό αποτέλεσμα είναι:



Στο confusion matrix αν πάρουμε την κύρια διαγώνιο του πίνακα είναι 459+5+0+…+0 = 464 σωστές προβλέψεις. Παρατηρούμε ότι ο dbscan έχει δημιουργήσει μόνο 2 συστάδες σε αντίθεση με τον k-means (στον οποίο δίναμε εμείς ως όρισμα τις συστάδες που θα δημιουργήσει αλλά όμοια και στον birch που θα δούμε παρακάτω)



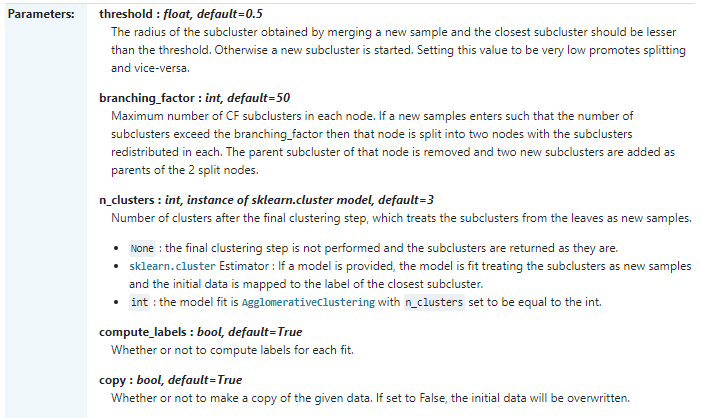


Τα παραπάνω διαγράμματα δείχνουν τις δύο συστάδες που σχηματίζονται με τον αλγόριθμο dbscan με διαφορετικό χρώμα για κάθε συστάδα τόσο σε δυσδιάστατο όσο σε τρισδιάστατο διάγραμμα.

Ο τρίτος και τελευταίος αλγόριθμος είναι ο birch.

Ο BIRCH (ισορροπημένη επαναληπτική μείωση και ομαδοποίηση με χρήση ιεραρχιών) είναι ένας μη εποπτευόμενος αλγόριθμος εξόρυξης δεδομένων που χρησιμοποιείται για την εκτέλεση ιεραρχικής ομαδοποίησης σε ιδιαίτερα μεγάλα σύνολα δεδομένων. Ένα πλεονέκτημα του BIRCH είναι η ικανότητά του να συγκεντρώνει σταδιακά και δυναμικά συγκεντρωτικά εισερχόμενα, πολυδιάστατα μετρικά σημεία δεδομένων σε μια προσπάθεια να παράγει ομαδοποίηση καλύτερης ποιότητας για ένα δεδομένο σύνολο πόρων (μνήμη και χρονικοί περιορισμοί). Στις περισσότερες περιπτώσεις, το BIRCH απαιτεί μόνο μία σάρωση της βάσης δεδομένων.

Το σύνολο των ορισμάτων (αλλά και ο ρόλος τους) που δέχεται σύμφωνα το scikitlearn είναι:



Αφήνουμε τα ορίσματα compute\_labels , bool ως έχουν με τις default τιμές τους μιας και δεν επηρεάζουν με κάποιο τρόπο την απόδοση του αλγόριθμου και συγκεκριμένα για το compute\_labels θα πρέπει να παραμείνει true ώστε να μπορούμε να έχουμε τα labels για να τα συγκρίνουμε αργότερα με τα πραγματικά labels.

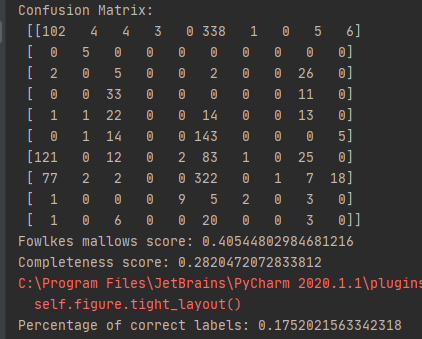
Tο n\_clusters το ορίζουμε 10 αφού έχουμε συνολικά 10 διαφορετικά labels CYT,NUC,MIT,ME3,ME2,ME1,EXC,VAC,POX,ERL

Οπότε τρέχουμε τον αλγόριθμο

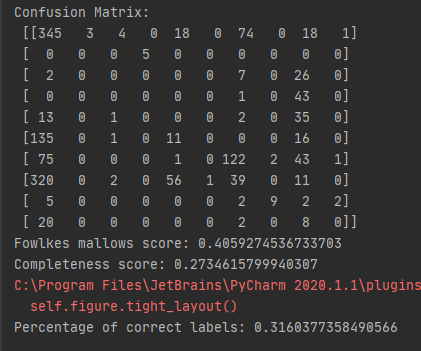
birch = Birch(n\_clusters=10,threshold=0.5,branching\_factor=50,compute\_labels=True,copy=True) # define the model



Για το όρισμα threshold αν το αυξήσουμε σε threshold=0.8

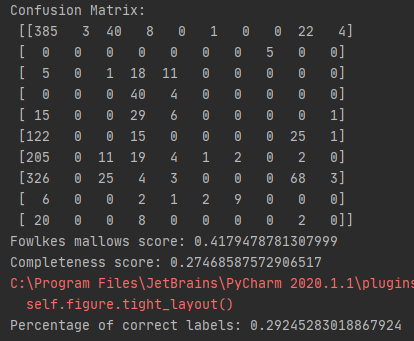


Η απόδοση βελτιώνεται. Αν το αυξήσουμε σε threshold=1.1



Η απόδοση βελτιώνεται ακόμα περισσότερο.

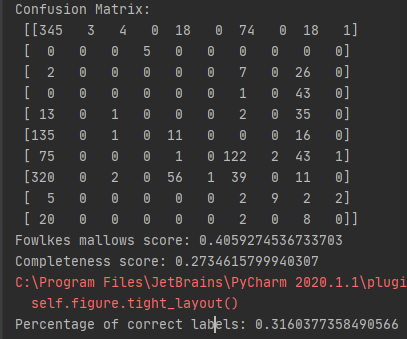
Για μεγαλύτερες τιμές από 1.1 η απόδοση αρχίζει να πέφτει πχ threshold=1.3



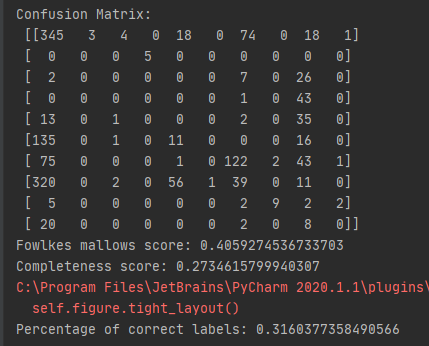
Οπότε κρατάμε threshold=1.1

Για το όρισμα branching\_factor δεν φαίνεται να επηρεάζει την απόδοση

Branching\_factor = 150

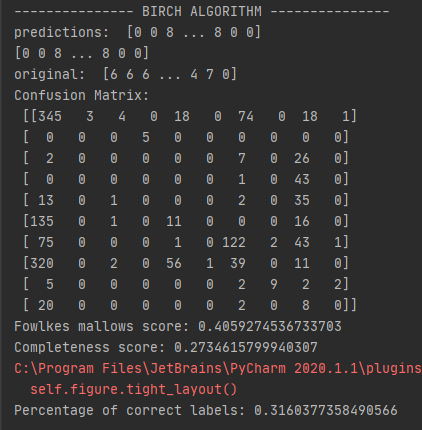


Branching\_factor = 20

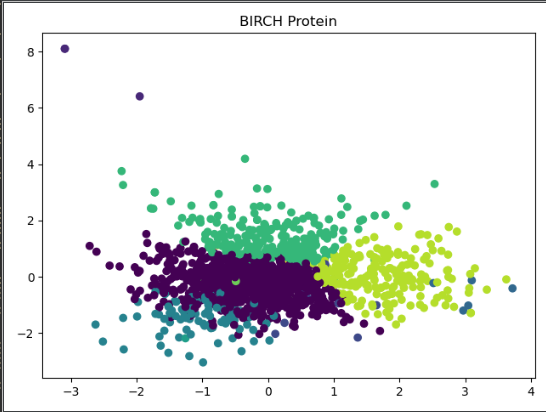


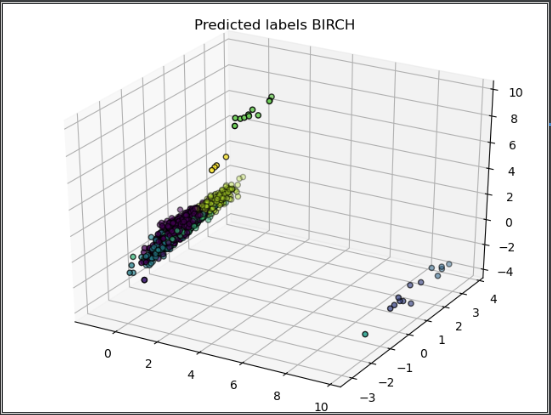
Οπότε το κρατάμε με την default τιμή του 50.

Το τελικό αποτέλεσμα που προκύπτει:



Όπου στο confusion matrix αν πάρουμε την κύρια διαγώνιο του πίνακα είναι 345+122+2+0+…+0 = 469 σωστές προβλέψεις ενώ σε όλα τα υπόλοιπα κελιά είναι ο λανθασμένες.

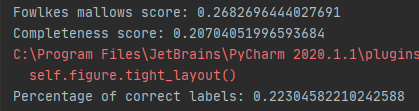




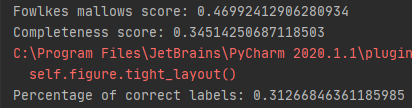
Τα παραπάνω διαγράμματα δείχνουν τις συστάδες που σχηματίζονται με τον αλγόριθμο birch με διαφορετικό χρώμα για κάθε συστάδα τόσο σε δυσδιάστατο όσο σε τρισδιάστατο διάγραμμα.

Συγκριτικά για και τους 3 αλγόριθμους έχουμε:

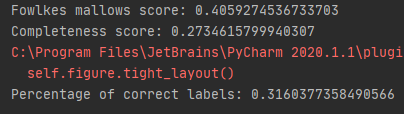
K-MEANS



DBSCAN



BIRCH



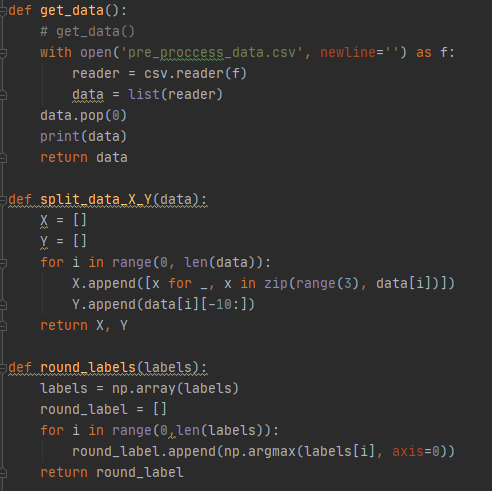
Οπότε σύμφωνα με τα metrics που παρέχει η scikit-learn ο πιο αποδοτικός αλγόριθμος ήταν ο DBSCAN ενώ αναλογικά με το πόσες σωστές προβλέψεις όπως είδαμε παραπάνω και με τα confusion matrix ήταν ο BIRCH με αμελητέα διαφορά. Ενώ ο χειρότερος από όλους ο K-MEANS.

Γενικά δεν υπάρχει κάποια στανταρ απάντηση για το ποιος αλγόριθμος είναι καλύτερος από τον άλλον αλλά εξαρτάται από την εφαρμογή γιαυτό και τους συγκρίνουμε πάνω στο κάθε πρόβλημα. Ο καθένας έχει θετικά και αρνητικά. Ο k-means τρέχει πολύ πιο γρήγορα από τον dbscan - brich αλλά ο dbscan δεν απαιτεί τον αριθμό των clusters όπως στον k-means – birch. Ο birch έχει 3 hyper-paramenters (number of clusters, threshold, and branching factor) πράγμα που καθιστά τη χρήση αυτού του αλγόριθμου αποτελεσματικά λίγο δύσκολη.

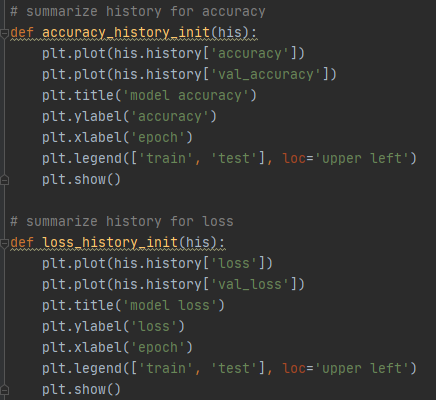
**1.3 classification – υλοποίηση με keras/tensorflow.**

Το αρχείο MLP.py υλοποιεί το ερώτημα 3.

Οι συναρτήσεις

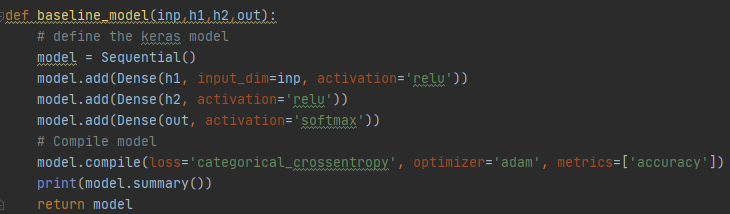


λειτουργούν με ακριβώς τον ίδιο τρόπο όπως περιγράφηκε και στο ερώτημα 2.

Οι παρακάτω 2 συναρτήσεις 

παράγουν τα διαγράμματα ιστορικού τιμών για την ακρίβεια αλλά και την απώλεια του μοντέλου όπως ζητείται στην εκφώνηση του ερωτήματος.

Τέλος , η συνάρτηση



δημιουργεί το μοντέλο και ορίζει ως activation functions στα δύο πρώτα στρώματα την relu και στο τελευταίο τη softmax.

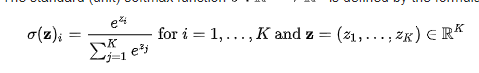
Η ReLu activation function φροντίζει το αποτέλεσμα να είναι στο πεδίο [0,άπειρο] και ο τύπος της είναι

**R(z)=max(0,z)**

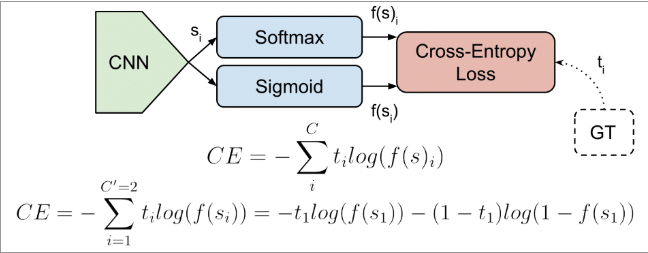
Με απλά λόγια αν βρεί αρνητικό αριθμό τον κάνει 0 αν είναι μεγαλύτερος του 0 τότε τον αφήνει όπως είναι.

Η Softmax activation function φροντίζει τα output να είναι από 0-1 και όλα μαζί να αθροίζουν σε 1 αντίθετα με την sigmoid η οποία δεν ικανοποιεί το δεύτερο αλλά μόνο το πρώτο. Δηλαδή το output θα είναι σε μορφή πιθανοτήτων για κάθε πιθανή κατηγορία.

O τύπος είναι ο παρακάτω:

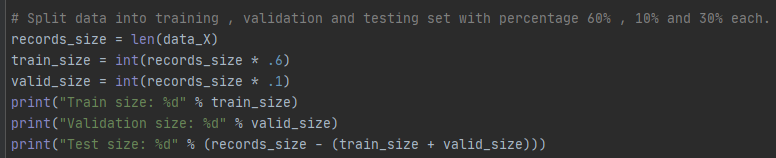


Για την απώλεια του μοντέλου χρησιμοποιείται η categorial crossentropy



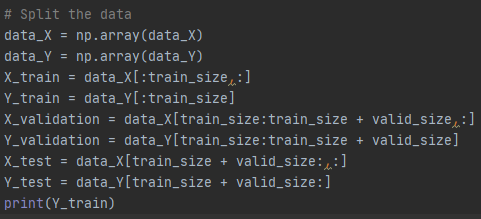
Αυτές είναι λοιπόν οι συναρτήσεις του προγράμματος.

Αρχικά στο πρόγραμμα υπολογίζονται τα μεγέθη του train , test , validation.

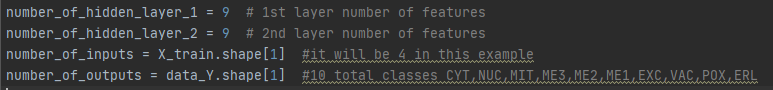


Για το train είναι το 60% του συνόλου των δεδομένων , για το validation είναι το 10% του συνόλου των δεδομένων ενώ για το test είναι ότι απομένει δηλαδή το 30%.

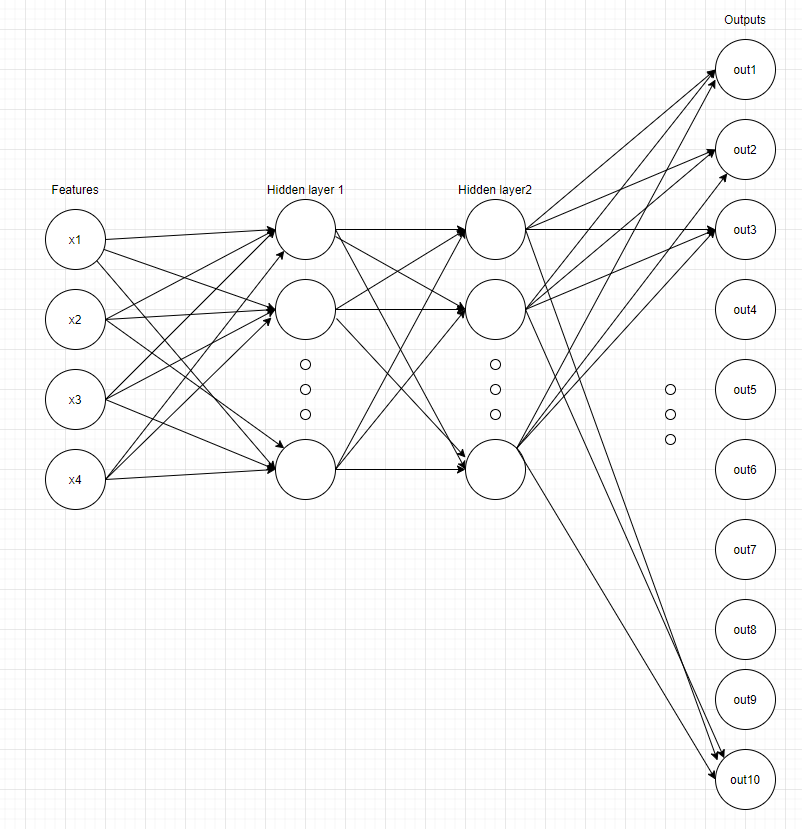
Στη συνέχεια με βάση τα παραπάνω μεγέθη σπάμε τα δεδομένα ανάλογα



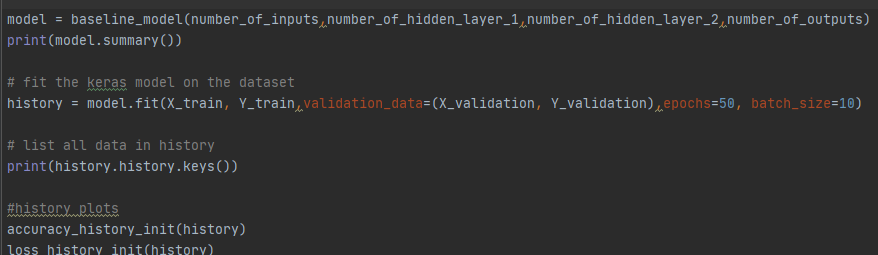
Έπειτα ορίζονται οι παράμετροι του νευρωνικού δικτύου.



Τα inputs γνωρίζουμε πως θα είναι 4 όπως και τα outputs 10 ενώ το ερώτημα ορίζει ότι το νευρωνικό δίκτυο θα πρέπει να έχει δύο κρυφά στρώματα. Θα είναι δηλαδή της μορφής:

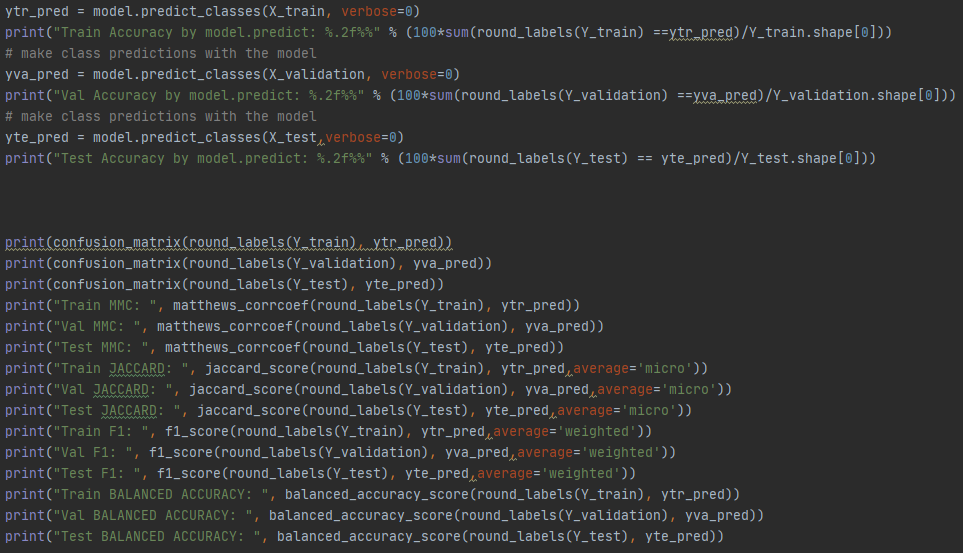


Μετά δημιουργείται το μοντέλο

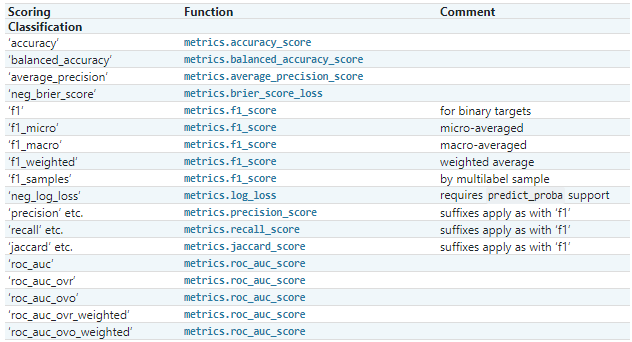


γίνεται η εκπαίδευση και εμφανίζονται τα διαγράμματα ιστορικού για την απώλεια και την ακρίβεια του μοντέλου όπως περιγράφηκε παραπάνω με τις αντίστοιχες συναρτήσεις.

Τέλος εμφανίζεται η ακρίβεια του δικτύου



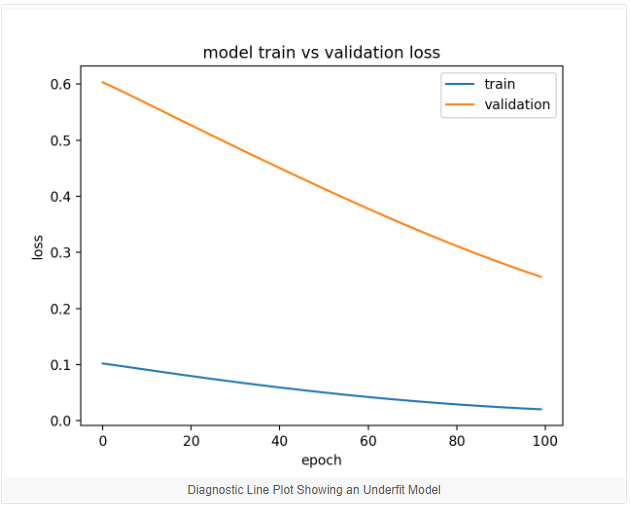
ενώ χρησιμοποιούνται διάφορες μετρικές που παρέχει η βιβλιοθήκη scikit-learn για προβλήματα classification



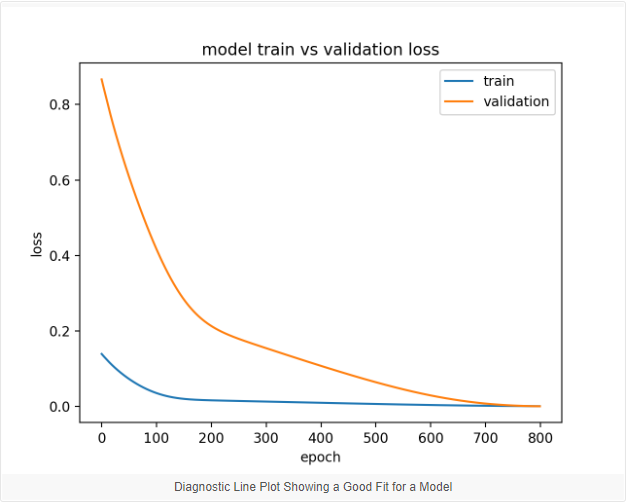
<https://scikit-learn.org/stable/modules/model_evaluation.html>

Η πληροφορία που μπορούμε να πάρουμε από τα διαγράμματα ιστορικού απώλειας είναι η εξής:

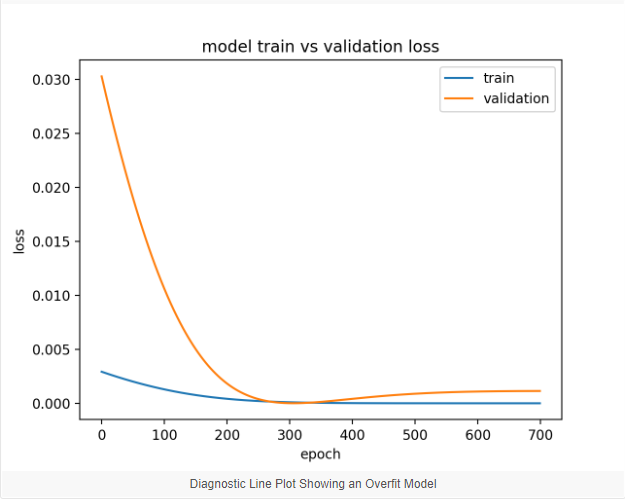
Ένα μοντέλο underfit είναι ένα μοντέλο που αποδεικνύεται ότι έχει καλή απόδοση στο σύνολο δεδομένων train και χαμηλό στο σύνολο δεδομένων validation. Αυτό μπορεί να διαγνωστεί από μια πλοκή όπου η απώλεια εκπαίδευσης είναι χαμηλότερη από την απώλεια επικύρωσης και η απώλεια επικύρωσης έχει μια τάση που υποδηλώνει ότι είναι πιθανές περαιτέρω βελτιώσεις. Για παράδειγμα θα μοιάζει κάπως έτσι



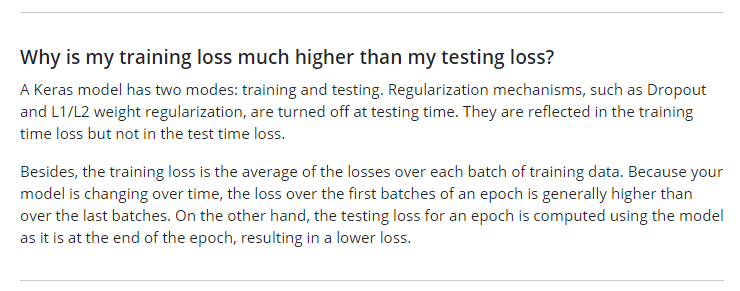
Η καλή εφαρμογή είναι μια περίπτωση όπου η απόδοση του μοντέλου είναι καλή τόσο στα σετ training όσο και στα σετ επικύρωσης. Αυτό μπορεί να διαγνωστεί από μια πλοκή όπου η απώλεια training και επικύρωσης μειώνεται και σταθεροποιείται γύρω στο ίδιο σημείο. Για παράδειγμα θα μοιάζει κάπως έτσι



Ένα μοντέλο overfit είναι αυτό όπου η απόδοση στο σετ training είναι καλή και συνεχίζει να βελτιώνεται, ενώ η απόδοση στο σετ επικύρωσης βελτιώνεται σε ένα σημείο και στη συνέχεια αρχίζει να υποβαθμίζεται.

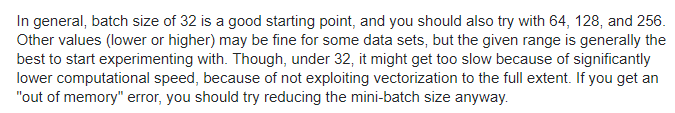


Ωστόσο στο keras μπορεί το validation loss να είναι μικρότερο του test loss , το λόγο που συμβαίνει αυτό μπορούμε να τον βρούμε στο documentation του keras



<https://keras.io/getting_started/faq/#why-is-my-training-loss-much-higher-than-my-testing-loss>

Όσο αφορά το όρισμα batch\_size



Επειδή ο όγκος των δεδομένων είναι μικρός ~1400 το ορίσουμε ως 25.

Το άλλο όρισμα είναι το epochs.

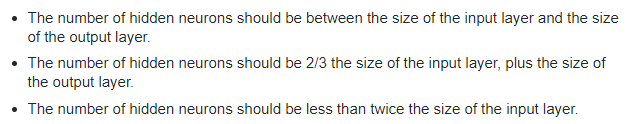
Ο αριθμός των εποχών θα αποφασίσει - πόσες φορές θα αλλάξουμε τα βάρη του δικτύου. Καθώς ο αριθμός των εποχών αυξάνεται, ο ίδιος αριθμός φορών που αλλάζουν τα βάρη στο νευρωνικό δίκτυο.

Για τον αριθμό των epochs τον ορίζουμε ως 100.

Ο καθορισμός του αριθμού των νευρώνων στα κρυμμένα στρώματα είναι ένα πολύ σημαντικό μέρος του καθορισμού της συνολικής αρχιτεκτονικής του νευρικού δικτύου. Αν και αυτά τα επίπεδα δεν αλληλεπιδρούν άμεσα με το εξωτερικό περιβάλλον, έχουν τεράστια επίδραση στην τελική παραγωγή. Τόσο ο αριθμός των κρυφών στρωμάτων όσο και ο αριθμός των νευρώνων σε καθένα από αυτά τα κρυμμένα στρώματα πρέπει να ληφθούν προσεκτικά υπόψη. Η χρήση πολύ λίγων νευρώνων στα κρυμμένα στρώματα θα έχει ως αποτέλεσμα το underfitting. Η υποσύνδεση συμβαίνει όταν υπάρχουν πολύ λίγοι νευρώνες στα κρυμμένα στρώματα για να ανιχνεύσουν επαρκώς τα σήματα σε ένα περίπλοκο σύνολο δεδομένων. Η χρήση πάρα πολλών νευρώνων στα κρυμμένα στρώματα μπορεί να οδηγήσει σε πολλά προβλήματα. Πρώτον, πάρα πολλοί νευρώνες στα κρυμμένα στρώματα μπορεί να οδηγήσουν σε overfitting. Η υπερφόρτωση συμβαίνει όταν το νευρικό δίκτυο έχει τόσο μεγάλη ικανότητα επεξεργασίας πληροφοριών που ο περιορισμένος όγκος πληροφοριών που περιέχονται στο σετ προπόνησης δεν επαρκεί για την εκπαίδευση όλων των νευρώνων στα κρυμμένα στρώματα. Ένα δεύτερο πρόβλημα μπορεί να προκύψει ακόμη και όταν τα δεδομένα εκπαίδευσης είναι επαρκή. Ένας υπερβολικά μεγάλος αριθμός νευρώνων στα κρυμμένα στρώματα μπορεί να αυξήσει το χρόνο που απαιτείται για την εκπαίδευση του δικτύου. Ο χρόνος εκπαίδευσης μπορεί να αυξηθεί στο σημείο που είναι αδύνατο να εκπαιδευτεί επαρκώς το νευρικό δίκτυο. Προφανώς, πρέπει να επιτευχθεί κάποιος συμβιβασμός μεταξύ πάρα πολλών και πολύ λίγων νευρώνων στα κρυμμένα στρώματα.

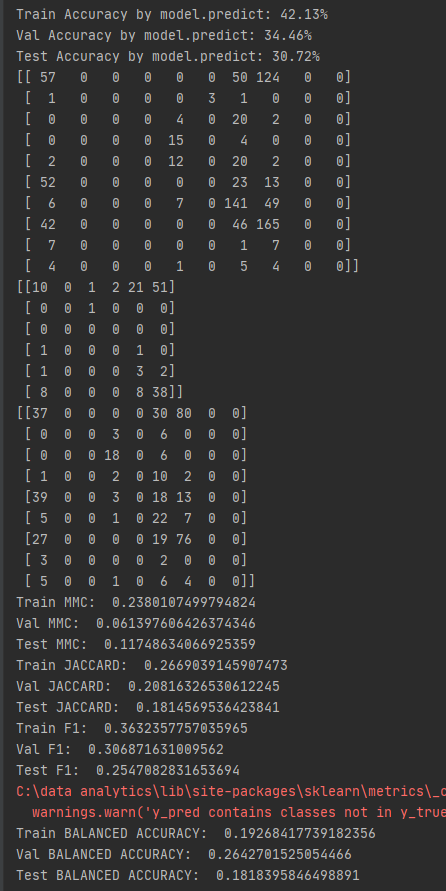
Αρχίζουμε τις δοκιμές για με τους αριθμούς νευρώνων για τα δύο hidden layer.

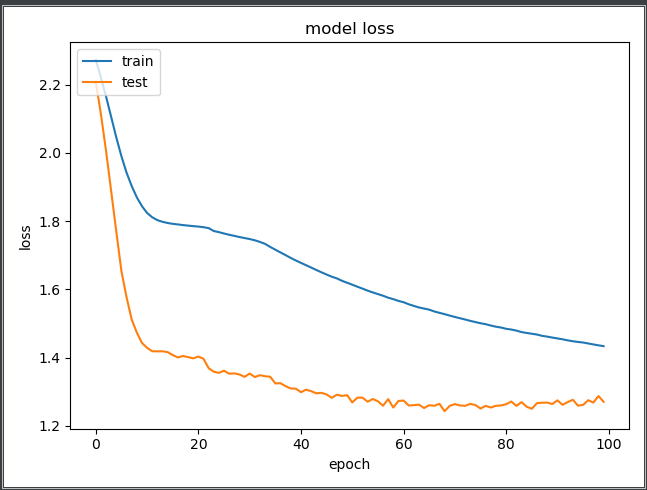
Υπάρχουν πολλές μέθοδοι για τον προσδιορισμό του σωστού αριθμού νευρώνων που θα χρησιμοποιηθούν στα κρυμμένα στρώματα, όπως τα ακόλουθα



Αυτοί οι τρεις κανόνες παρέχουν μια καλή αρχή για να λάβουμε υπόψη. Τελικά, η επιλογή μιας αρχιτεκτονικής για το νευρωνικό δίκτυο θα καταλήξει με βάση τις δοκιμές και τα σφάλματα.

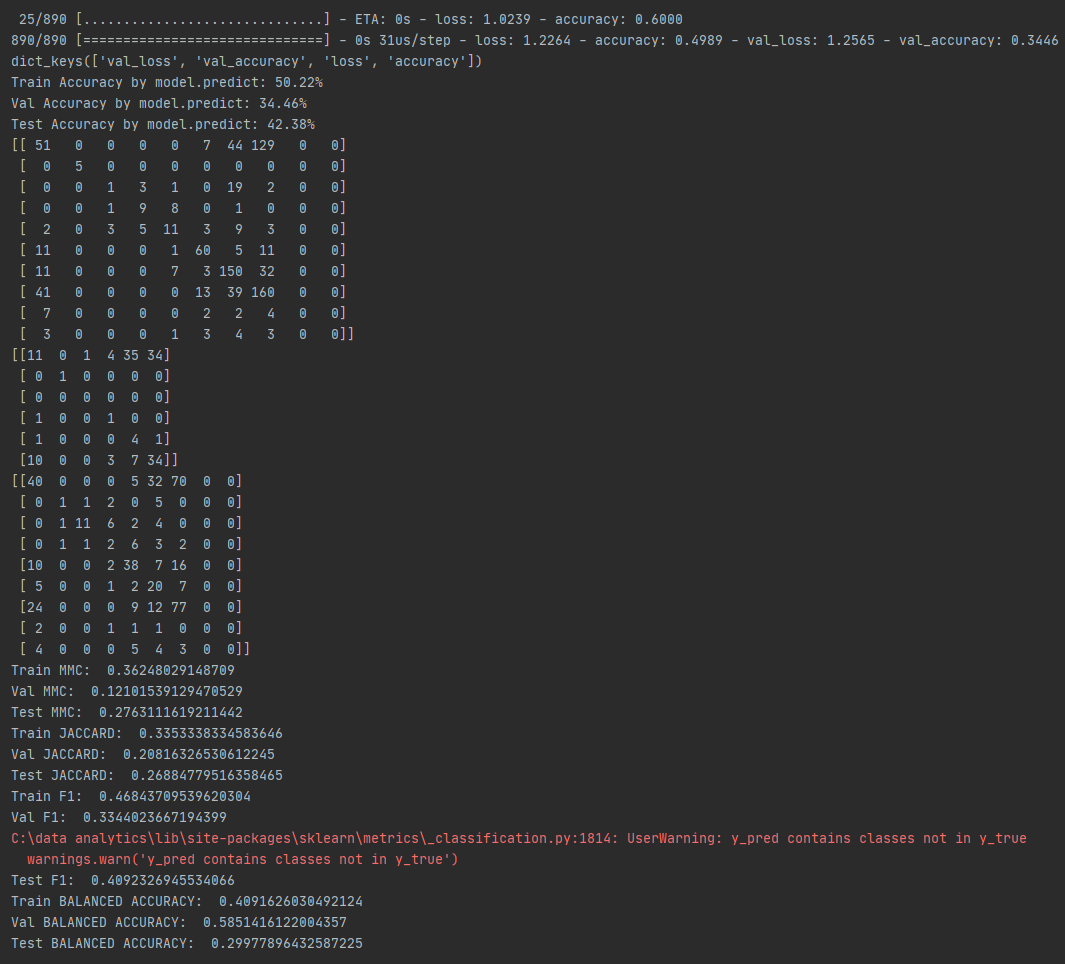
Έστω ότι έχουν από 5 νευρώνες

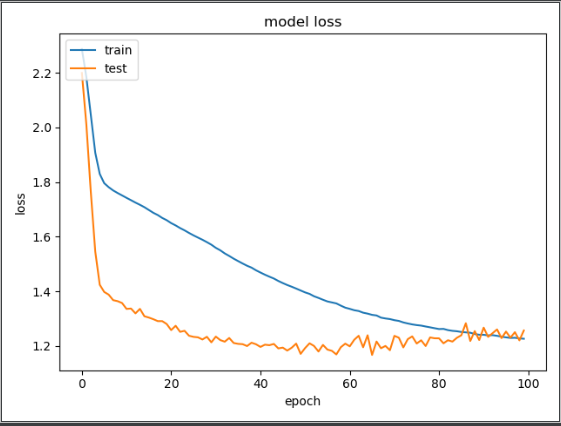




Στο οποίο είναι φανερό ότι υπάρχει underfitting.

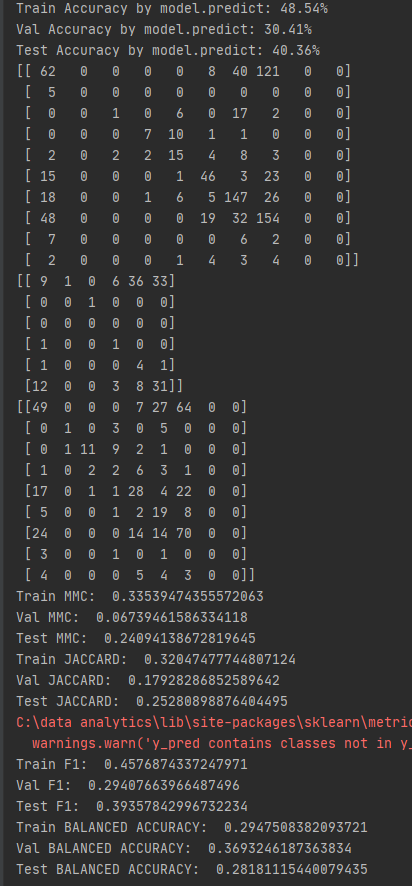
Γίνεται αύξηση από 5 σε 10.

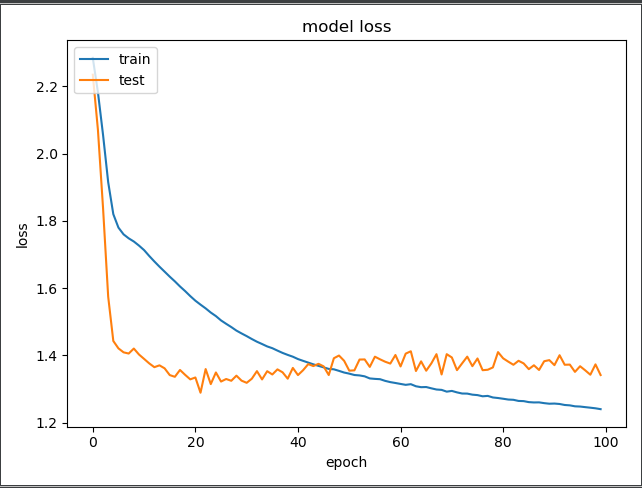




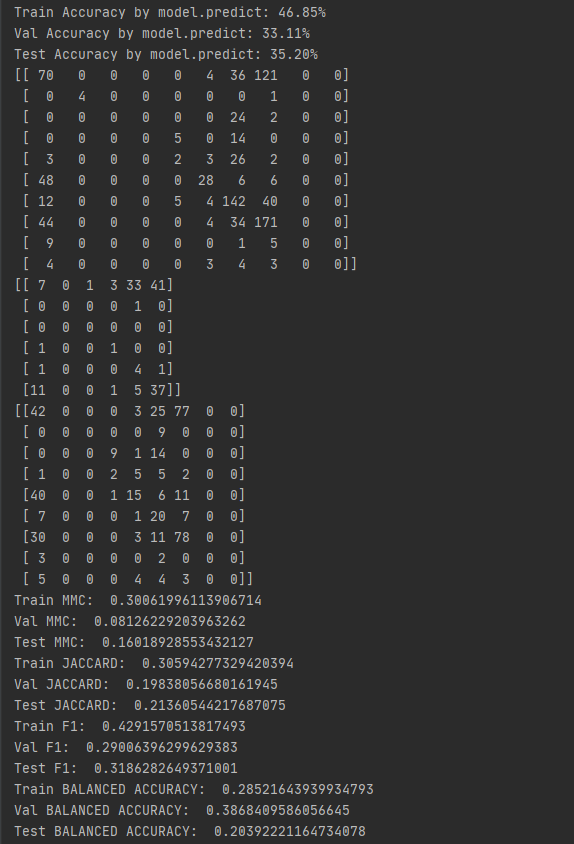
Τώρα το πρόβλημα παύει να υπάρχει.

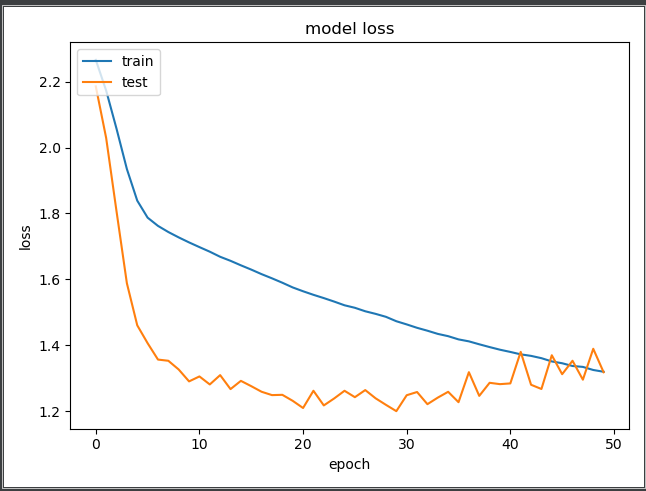
Αν τώρα το αυξήσουμε σε 15 νευρώνες σε κάθε hidden layer.



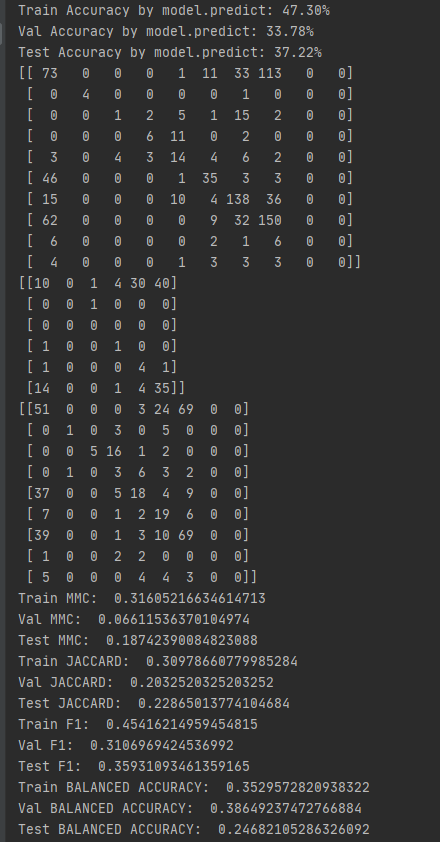


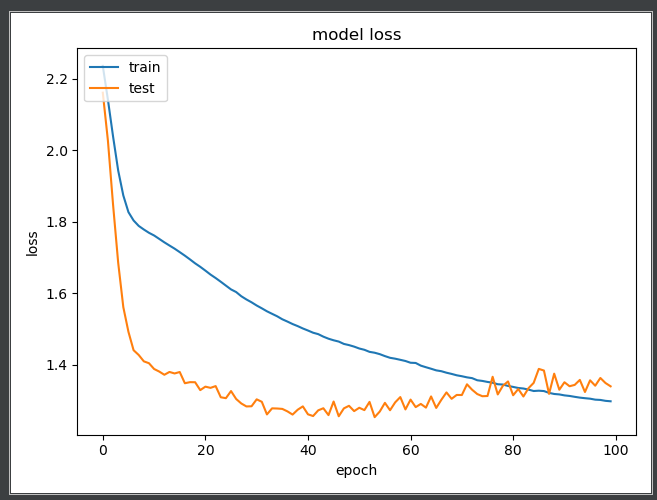
Τώρα υπάρχει πρόβλημα overfitting και όπως φαίνεται θα πρέπει να πέσει το epoch κάπου στο μισό του 40-60 οπότε έστω ότι το ορίζουμε 50.





Οπότε κρατάμε 10 νευρώνες για κάθε hidden layer. Αν δοκιμάσουμε να μειώσουμε τους νευρώνες για ένα από τα δύο hidden layer πχ 10 για το ένα και 8 για το άλλο



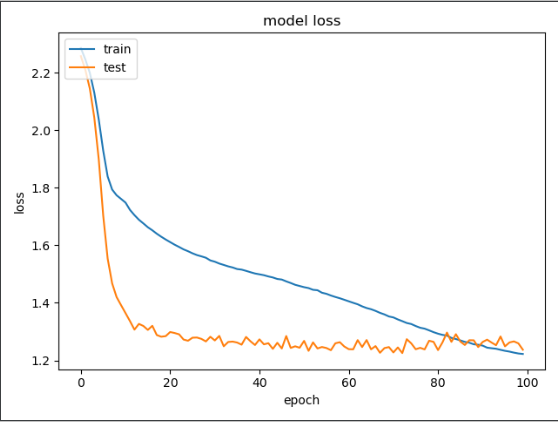


Τότε σύμφωνα με τους στατιστικούς δείκτες και την ακρίβεια του δικτύου έχουμε χειρότερα αποτελέσματα το ίδιο συμβαίνει και αν αυξήσουμε λίγο για ένα από τα δύο hidden layers.

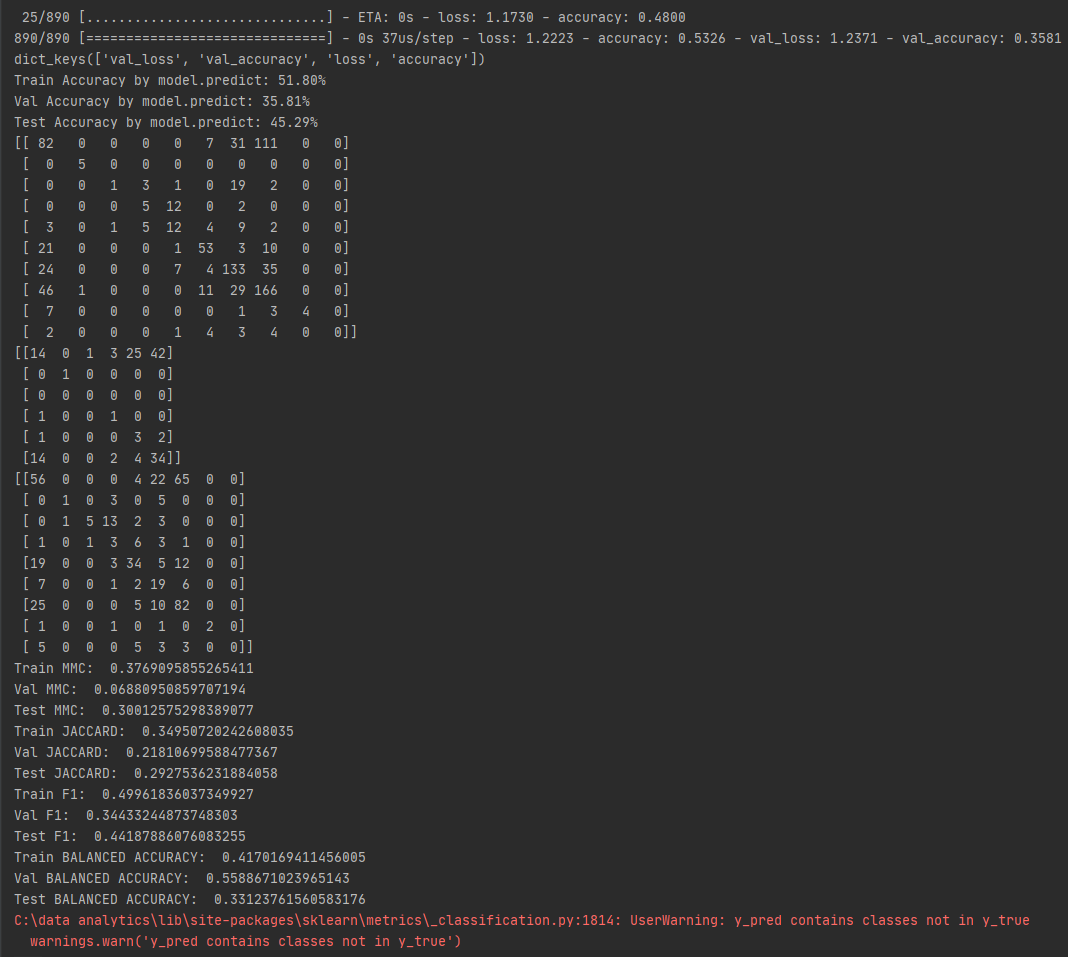
**Όσο για την αλλαγή του αριθμού των νευρώνων και την μεταβολή** **της ακρίβειας** και των στατιστικών δεικτών όπως φαίνεται στις παραπάνω φωτογραφίες από 5 νευρώνες σε 10 υπήρξε αρκετά μεγάλη αύξηση στους στατιστικούς δείκτες ενώ σε όλες τις άλλες περιπτώσεις υπήρξε πάρα πολύ μικρή αυξομείωση ή overfitting.

Γενικά όσο αυξάνονται οι νευρώνες στα hidden layers το μοντέλο γίνεται πιο προσαρμοστικό, ώστε να μπορεί να μάθει μικρότερες λεπτομέρειες. Αλλά αυτό δεν σημαίνει ότι είναι απαραίτητο, ότι ο ταξινομητής είναι τότε καλύτερος στο «επόμενο σύνολο δεδομένων». Το μοντέλο επηρεάζεται περισσότερο από το over-fitting και έτσι η γενίκευση του μοντέλου ταξινόμησης μπορεί επίσης να μειωθεί, π.χ. ο ταξινομητής θα λειτουργήσει χειρότερα στο επόμενο σύνολο δεδομένων.

Τελικό output προγράμματος







Παραπάνω φαίνονται τα ποσοστά ακρίβειας κατά το training , validation , testing καθώς τα confusion matrix (στην κύρια διαγώνιο φαίνονται των πινάκων φαίνονται οι σωστές προβλέψεις του δικτύου) αλλά και οι στατιστικοί δείκτες και για τα τρία.

Όπως φαίνεται η ακρίβεια για το validation είναι μικρότερη του testing αλλά η ακρίβεια του training πολύ λίγο μεγαλύτερη ~6%. Γενικά το επιθυμητό είναι το accuracy για το testing set να είναι το μεγαλύτερο.

Ωστόσο , αυτό μπορεί να συμβεί εάν η επικύρωσή έχει οριστεί κατά σειρά μεγέθους μικρότερο από το training set. Σε αυτήν την περίπτωση, το σύνολο επικύρωσης μπορεί να μην καλύπτει όλες τις πολλαπλές πληροφορίες εκπαίδευσης, αλλά μόνο κάποιο τοπικό μέρος αυτών. Επομένως, κατά λάθος, ο ταξινομητής λειτουργεί καλύτερα από τον μέσο όρο σε όλα τα σύνολα δεδομένων.

**2. ΜΕΡΟΣ Α**

**2.1 Προπαρασκευή δεδομένων – υλοποίηση με εργαλεία στατιστικής επεξεργασίας, π.χ. Excel, R/Python.**

## Ερώτημα 1

Η επίλυση του ερωτήματος 1 βρίσκεται στο αρχείο preprocessing.py . Ακολουθεί φωτογραφία του κώδικα και η επεξήγηση:

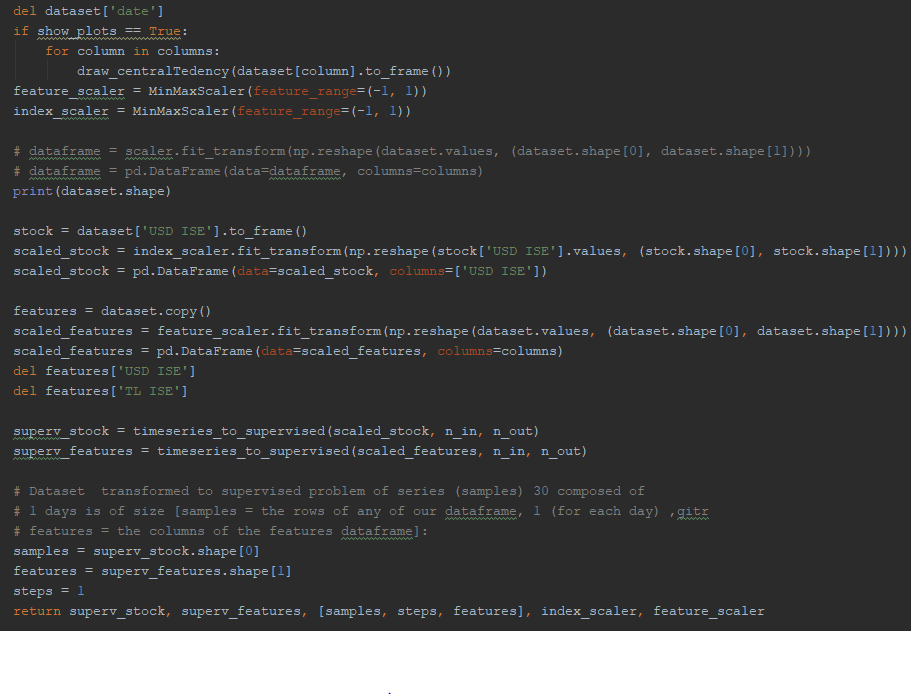
Η προεπεξεργασία των δεδομένων εκτελείται μέσω της συνάρτησης preprocess.Η συνάρτηση preprocess δέχεται ως ορίσματα τις μεταβλητές dataset (τύπου pandas Dataframe), n\_in, n\_out (αντιπροσωπεύουν τον αριθμό των lag observations τύπου integer) και της μεταβλητή show\_plots(τύπου Boolean).

Aρχικά κάνουμε reset τα indexes του dataframe ώστε να ξεκινάνε απο το 0 και κάνουμε drop την έξτρα στήλη που δημιουργείται.Επιλέγουμε τα πρώτα 530 στοιχεία και εκχωρούμε στο dataframe τις σωστές

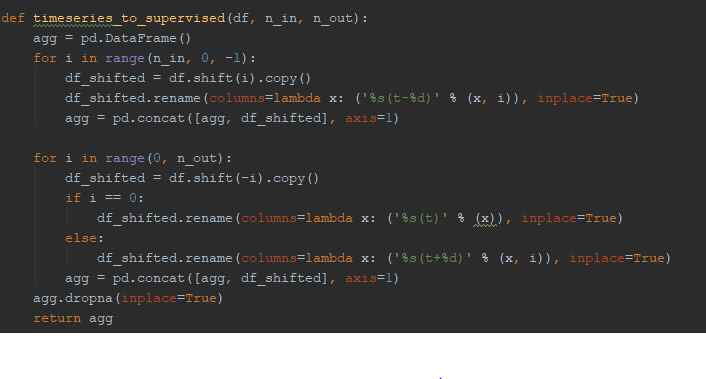
στήλες.Αν η μεταβλητή show\_plots έχει την τιμή True τότε εμφανίζουμε διαγράμματα τύπου bar plot, lag plot, autocorrelation plot, scatter plot κλπ ώστε να κατανοήσουμε καλύτερα τα δεδομένα μας και να αποκτήσουμε κάποια ενόραση σχετικά με την δομή τους, την κατανομή τους κλπ



Στην συνέχεια διαγράφουμε την στήλη date και κάνουμε scale τα δεδομένα μας στο μήκος -1 εως 1 με τον MinMaxScaler. Διαγράφουμε περιττές στήλες απο τα features και μετατρέπουμε το πρόβλημα απο time series σε supervised learning με την χρήση της **timeseries\_to\_supervised** την οποία θα επεξηγήσουμε παρακάτω.Έπειτα μετατρέπουμε το πρόβλημα στην μορφή [samples, features, steps] όπου samples είναι οι διάσταση των σειρών οποιουδήποτε απο τα dataframe superv\_stock, superv\_features καθώς τα rows των δεδομένων μας είναι ίδια και features είναι η διάσταση των στηλών του dataframe superv\_features.Και τα steps παίρνουν την τιμή 1. **Η επιλογή έγινε με βάση το ότι το dataset αποτελείται απο μοναδικές ημερομηνίες (datetime) όπου στην κάθε μια αντιστοιχούν features διαφορετικής τιμής. Άρα κάθε μέρα διαφορετικό sample**.Τέλος η συνάρτηση επιστρέφει τα dataframe που κάναμε scale, λίστα με τα samples, steps, features καθώς και scaler objects καθώς θα χρειαστούν στην συνέχεια.

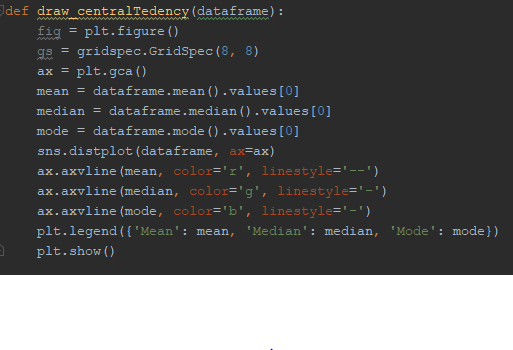


**Η** **timeseries**\_to\_supervised:

Δέχεται ένα dataframe και τα lag observations n\_in, n\_out ( n\_in lag obs για το παρελθόν (t-1, t-2 ….) n\_out lag obs για το μέλλον (t+1, t+2 …).Το dataframe γίνεται shift ανάλογα με τις περιόδους που του δίνονται. Κάνουμε drome null τιμές και το επιστρέφουμε.

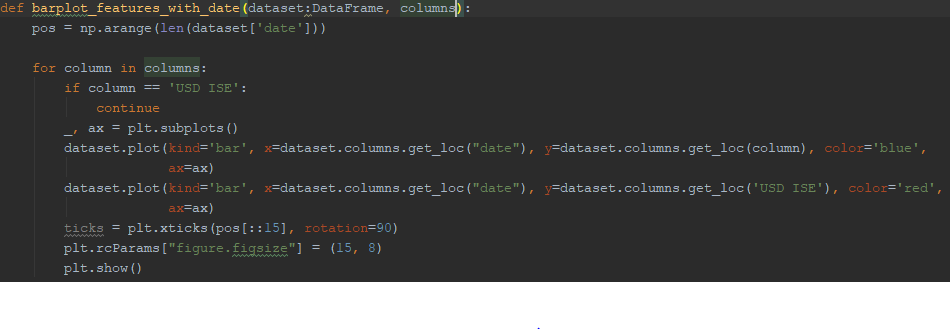
**Η draw\_centralTedency:**

Δέχεται dataframe και κάνει plot την κεντρική τάση (mean, mode, median).



Η **barplot\_features\_with\_date:**

Δέχεται ένα dataframe και τα ονόματα των στηλών.Στην συνέχεια δημιουργεί ενα κοινό barplot για τον δείκτη USD ISE και οποιονδήποτε άλλο υπάρχει στην μεταβλητή columns στον άξονα y σε σχέση με την ημερομηνία στον άξονα x.

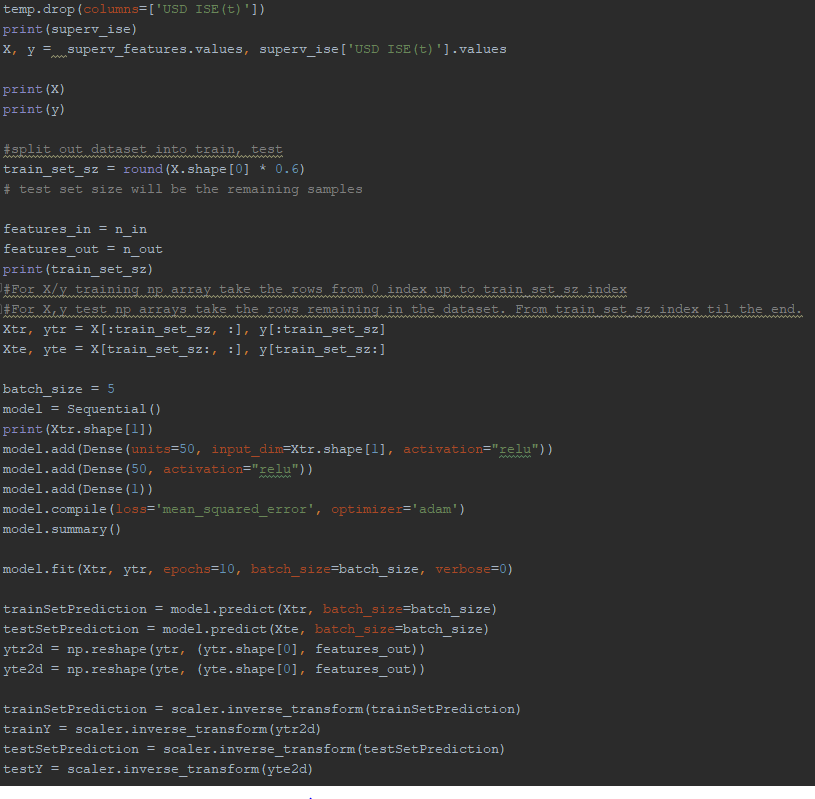


## Ερώτημα 2 α)

Στο ερώτημα αυτό δεν χρησιμοποιήσαμε διαγράμματα καθώς είναι ίδια με αυτά του ερωτήματος **2 γ)** του οποίου η ανάλυση θα γίνει στην συνέχεια.



Αρχικά εισάγουμε τα δεδομένα μας απο το αρχείο excel που έχουν αποθηκεύτει. Διαγράφουμε null τιμές, επιλέγουμε τα πρώτα 530 (κάνουμε skip τα rows με τα headers) , διαγράφουμε αχρείαστες στήλες και τα κάνουμε scale στο εύρος -1 εως 1. Θέτουμε τα lag observations ίσα με 1 καθώς μέσω δοκιμών η επιβολή υψηλότερης τιμής δεν επηρέαζε το αποτέλεσμα σημαντικά.Τέλος μετατρέπουμε το πρόβλημα απο timeseries σε supervised learning.



Στο vector Χ θα βάλουμε τα features μας ( δείκτες SP, NIKKOI , κλπ). Καθώς βάση του παρακάτω paper:  **Akbilgic, O., Bozdogan, H., Balaban, M.E., (2013) A novel Hybrid RBF Neural Networks model as a forecaster, Statistics and Computing. DOI 10.1007/s11222-013-9375-7**

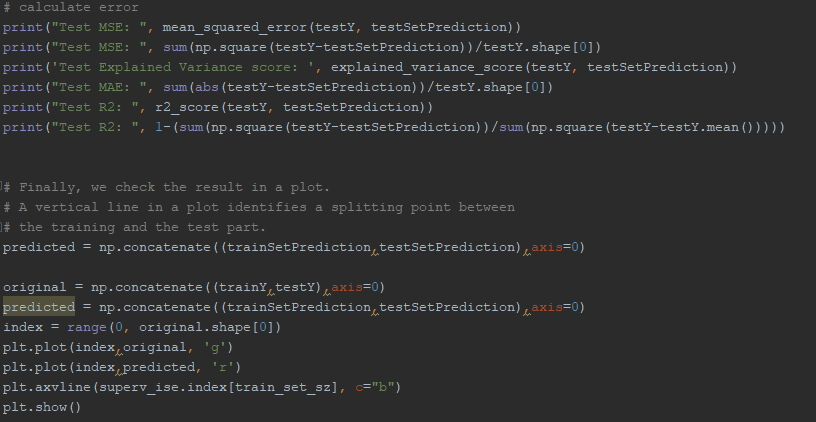
**Link:** <https://www.researchgate.net/publication/256091218_A_novel_Hybrid_RBF_Neural_Networks_model_as_a_forecaster>

Όλοι αυτοί οι δείκτες έχουν χρησιμοποιηθεί με διαφορετικούς συνδιασμούς για την πρόβλεψη του δείκτη ISE.Δεν θα χρησιμοποιήσουμε τα (παρελθοντικά (t-1 , t-2....) ) lag observations του δείκτη ISE. **Η αιτιολόγηση θα γίνει στο ερώτημα 2 κατα την παρουσίαση των διαγραμμάτων των δεδομένων.**

Στο vector y τοποθετούνται οι τιμές του δείκτη που θέλουμε να προβλέψουμε( USD BASED ISE).

Θέτουμε το μέγεθος του training set ίσο με το 60% του dataset και το μέγεθος του test set ίσο με 40% .

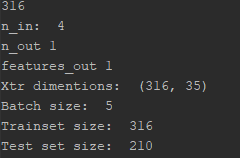
Έπειτα δημιουργούμε το (sequencial) μοντέλο του νευρωνικού δικτύου μας όπου το input layer έχει τόσα inputs όσα και τα feature μας. Δύο hidden layers με 50 νευρώνες το καθένα και τέλος το output layer με ένα output.Κάνουμε compile και fit το μοντέλο μας πάνω στα training set για τα vectors X,y (xtr, ytr).Κάνουμε τα predictions και στην συνέχεια μετατρέπουμε τα Y σε numpy arrays ώστε να κάνουμε reverse το scaling.

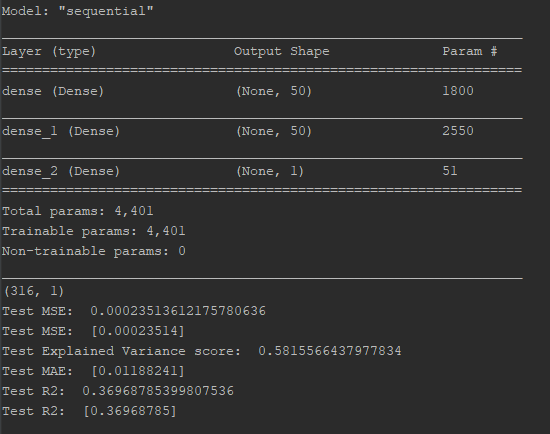


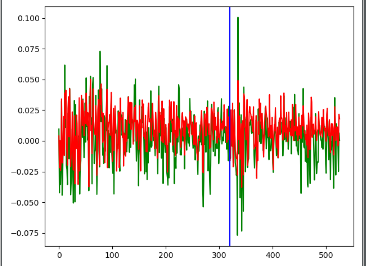
Τέλος υπολογίζουμε τα MSE, MAE, R2, explained variance scores και κάνουμε plot σε διάγραμμα κατά πόσο τα predictions μας ταιριάζουν στις original τιμες μας.

**2.2 Time-series prediction – υλοποίηση με keras/tensorflow**

## Επίδειξη κώδικα ερωτήματος 2 α)





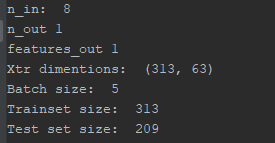


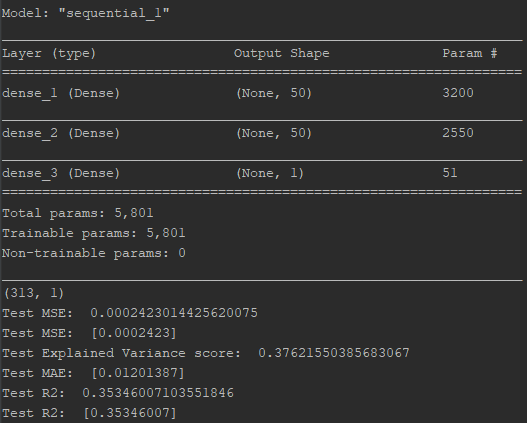
Παρατηρούμε ότι τα score των **MSE, MAE** είναι αρκετά μικρά ( κοντά στο 0 ) το οποίο είναι επιθυμητό καθώς πρεπει να ελαχιστοποιούνται αφού μας δείχουν το πόσο κοντά είναι η προβλέψεις μας στις πραγματικές τιμές.

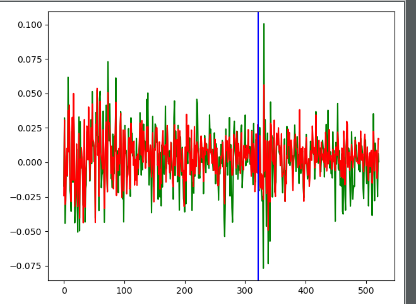
To R2 score δείχνει το πόσο καλά γίνεται fit το μοντέλο μας (<https://www.researchgate.net/post/What_does_the_R_square_coefficient_mean_and_why_is_there_different_value_when_computing_with_Excel>). Η τιμή που επιστρέφεται είναι μικρότερη του 0.6 οπότε δεν θεωρείται μια καλή και αποδεκτή τιμή. (<https://www.researchgate.net/post/What_is_the_acceptable_R-squared_in_the_information_system_research_Can_you_provide_some_references>).

Το explained variance score επίσης είναι μικρότερο του 0.6.Όσο μεγαλύτερο το score τόσο καλύτερη η συσχέτιση μεταξύ των δεδομένων μας καθώς και επίσης τόσο καλύτερες και οι προβλέψεις μας. (<https://www.statisticshowto.com/explained-variance-variation/>)

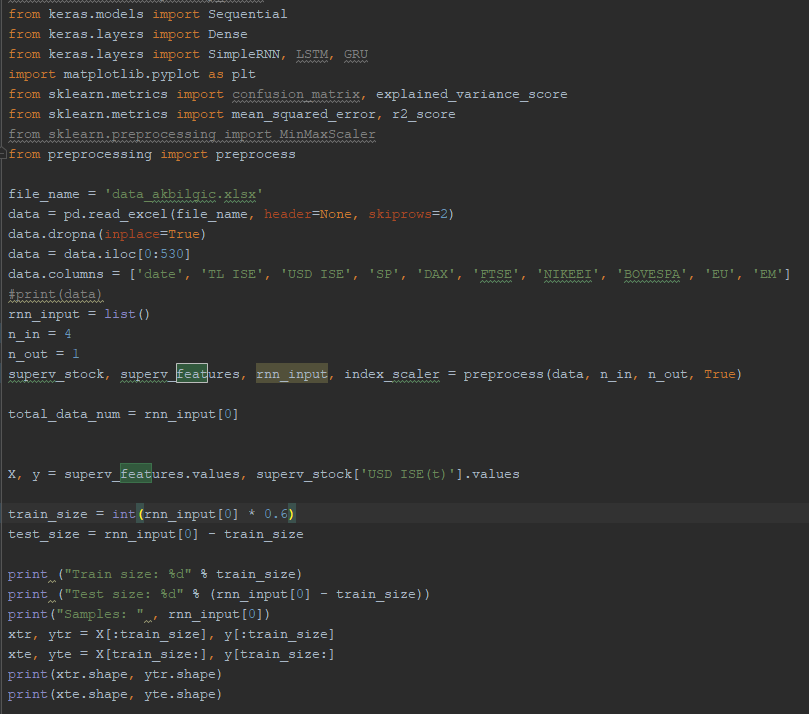
Στο παραπάνω plot με πράσινο έχουμε τις πραγματικές τιμές του δείκτη USD BASED ISE ενώ με κόκκινο τις προβλέψεις που κάναμε. Παρατηρούμε ότι σε μεγάλο βαθμό οι προβλέψεις μας ταίριαζαν με τις πραγματικές τιμές παρόλαυτα δεν θα μπορούσαμε να πούμε ότι το μοντέλο μας είναι πολύ καλό.Αυτό οφείλεται σε διάφορους λόγους. Ένας απο αυτούς είναι τα ίδια μας τα δεδομένα τα οποία δεν είναι αρκετά σε πλήθος ώστε το μοντέλο μας να μπορεί να κάνει γενικεύσεις. Επίσης παρατηρήθηκε ότι όσο περισσότερο αυξάναμε τα lag observations τόσο περισσότερο δεν ταίριαζαν οι προβλέψεις με τις πραγματικές τιμές και τα R2, explained variance score μειωνόντουσαν.Περισσότερες εξηγήσεις θα δωθούν παρακάτω κατά την παρουσίαση των διαγραμμάτων των δεδομένων τα οποία είναι κοινά και για τα ερωτήματα 2 a) , 2 b).Παρόλαυτα αξίζει να σημειωθεί ότι δίνοτας μεγάλη τιμή στην παράμετρο n\_in κάποιες φορές τα αποτελέσματα που πέρναμε ήταν λιγο έως και πολύ χειρότερα όσο αυξάνονταν το n\_in. Δηλαδή:



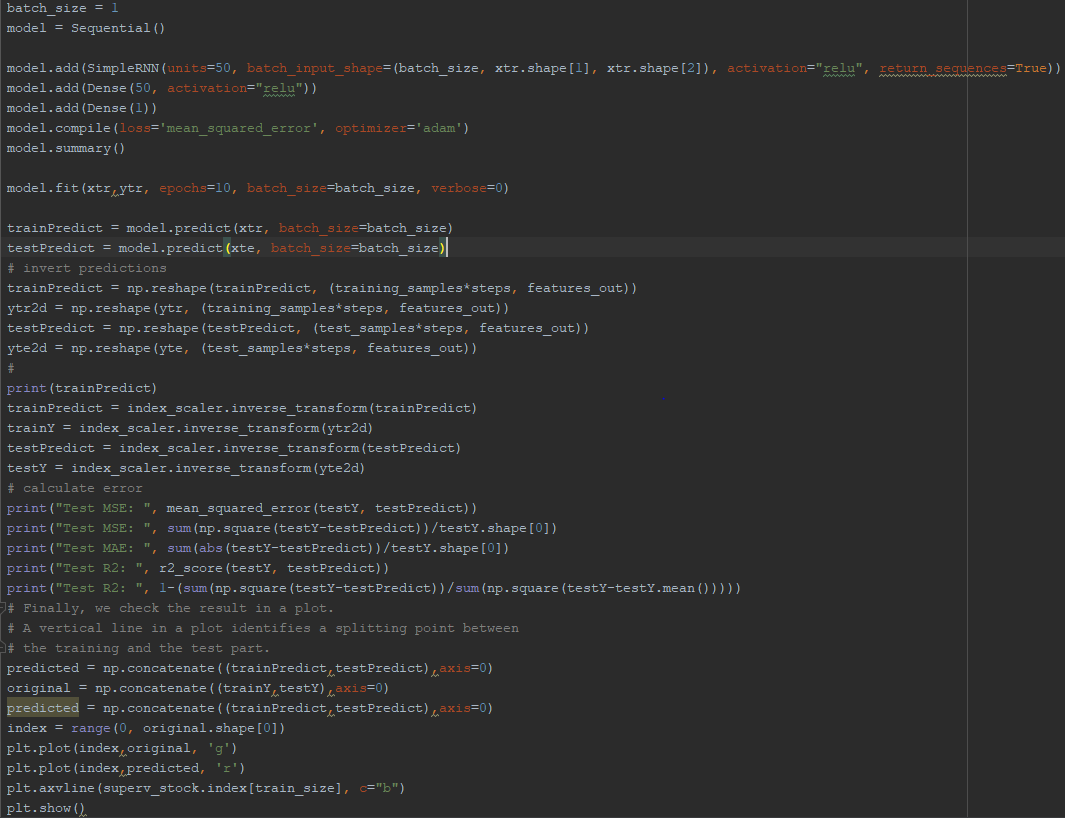




## Ερώτημα 2 b)

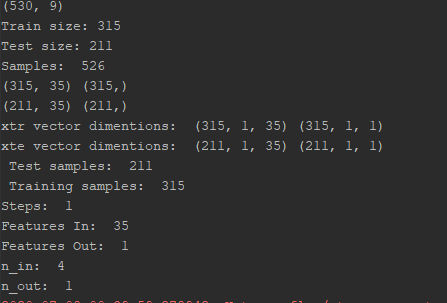


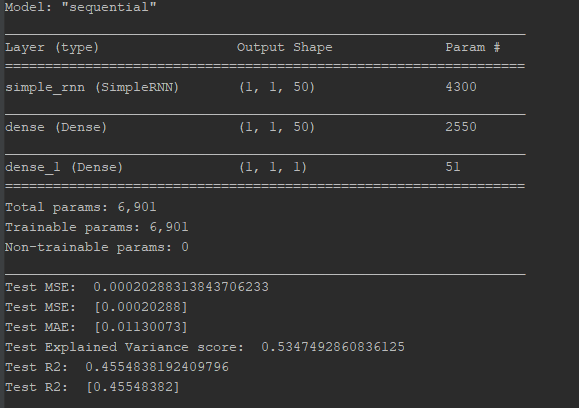
Όπως και στο προηγούμενο ερώτημα διαβάζουμε τα δεδομένα μας απο το excel αρχείο που είναι αποθηκευμένα.Αυτή την φορά όμως καλούμε την συνάρτηση **preprocess** την οποία επεξηγήσαμε προηγουμένως ώστε να γίνει η κατάλληλη προεπεξεργασία στα δεδομένα και να έρθει στην σωστή μορφή ώστε να είναι απο time series σε supervised learning. Επίσης το πρόβλημα μετατρέπεται στην μορφή [samples, steps, features].(Κάθε ημέρα αποτελεί διαφορετικό sample όπως το παράδειγμα στις διαφάνειες).Στην συνέχεια η διαδικασία που ακολουθείται είναι παρόμοια με αύτη του ερωτήματος 2 a). Τοποθετούμε τα δεδομένα μας στα vectors X, y. Χωρίζουμε το dataset σε train και test. (60% train 40% test) Και τα φέρνουμε στην κατάλληλη μορφή ώστε να χρησιμοποιηθούν στην συνέχεια.

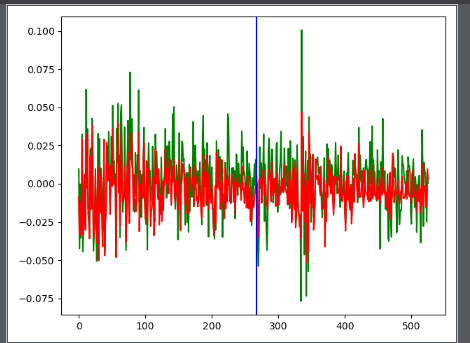


Έπειτα δημιουργούμε το μοντέλο μας και του δίνουμε τις κατάλληλες παραμέτρους. Πιο συγκεκριμένα το input layer θα δημιουργηθεί μέσω του batch\_input\_shape όπου έχει την μορφή [batch\_Size, timesteps, features].Το batch size το ορίζουμε εμείς στην προκειμένη περίπτωση το αφήσαμε 1. Τα timesteps, features παίρνουν τις τιμές των διαστάσεων του πίνακα xtr. (timestep = xtr.shape[1], features = xtr.shape[2].)Το activation function όπως και στο προηγούμενο ερώτημα είναι το relu, και έχουμε True στο return\_sequences (επιστρέφεται το output του hidden state κάθε input time step).Στην συνέχεια δημιουργούμε 2 hidden layer 50 νευρωνων και τέλος έχουμε 1 output layer. Κάνουμε fit το μοντέλο μας και ακολουθούμε την ίδια διαδικασία με το προηγούμενο ερώτημα. Διαφοροποίηση υπάρχει στο σημείο που κάνουμε reshape τα δεδομένα μας όπου χρησιμοποιούμε στις διαστάσεις τις τιμες των samples, steps, features\_out. Τέλος υπολογίζουμε τα score μας.

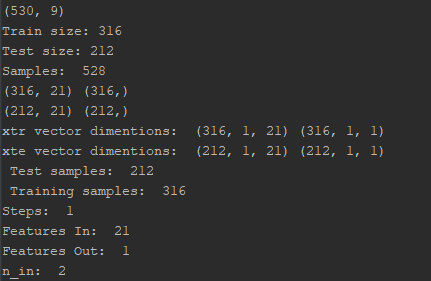
## Επίδειξη κώδικα ερωτήματος 2 β)

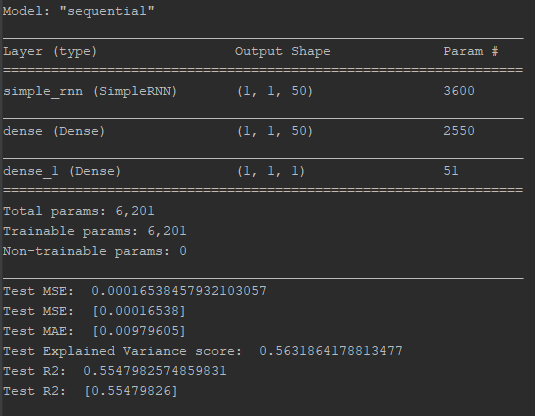


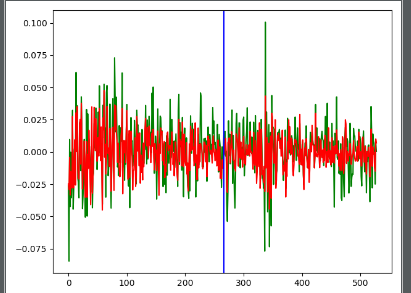




Για λιγότερα lag observations (n\_in):







Παραπάνω φαίνονται τα **steps, samples, features** κλπ που χρησιμοποιήθηκαν.Επίσης τα score του μοντέλου μας. Παρατηρούμε ότι και αυτή την φορά τα αποτελέσματα δεν είναι και τόσο υψηλά.Πιο συγκεκριμένα τα **MSE, MAE scores είναι οπως και πριν χαμηλά, όμως έχουμε χαμηλά R2, explained variance error scores.** Αυτό είναι εμφανές και από το διάγραμμα των προβλεπόμενων και αληθινών τιμων που βρίσκεται παραπάνω. Οι προβλεπόμενες τιμές (κοκκινο) βλέπουμε ότι ταιριάζουν μόνο σε μερικά σημεία με τις πραγματικές τιμές (πράσινο**). Έχουμε όμως λίγο καλύτερα score από τα MLP.Αυτό οφείλεται στο γεγονός ότι τα RNN «θυμούνται» προηγούμενα inputs (τιμές) ( Long Short Term Memory) με αποτέλεσμα να είναι περισσότερο αποτελεσματικά σε προβλήματα σχετικά με time series prediction.Ένας ακόμη λόγος που τα μοντέλα μας δίνουν χαμηλά R2, Explained Variance score είναι πως το dataset δεν είναι αρκετά μεγάλο.**

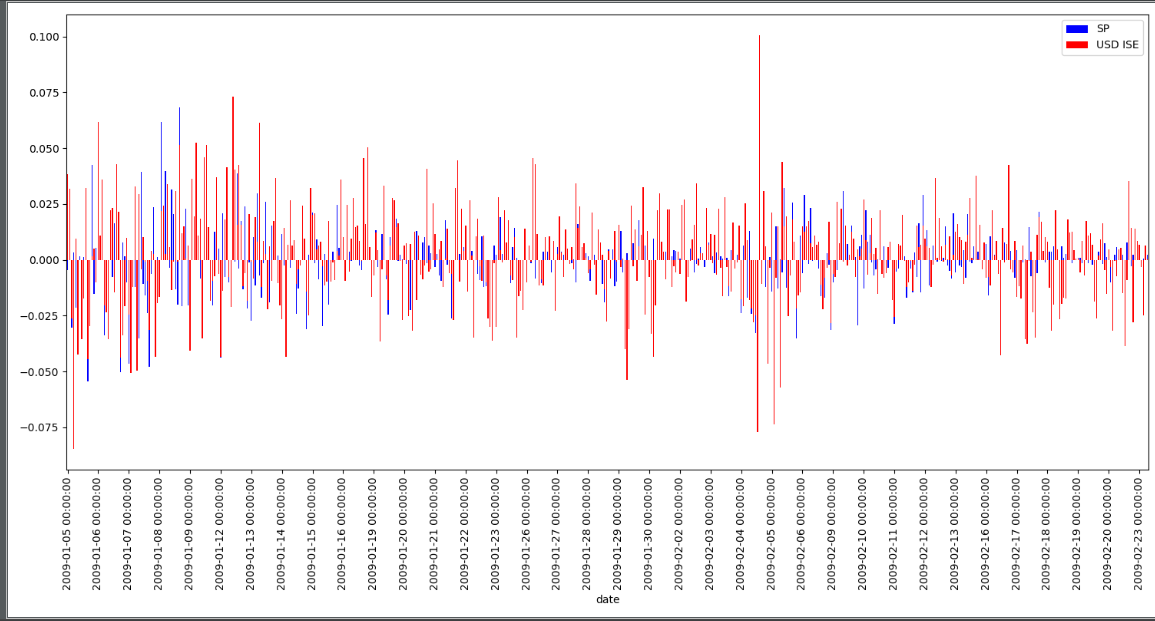
Όπως θα αναφερθεί και παρακάτω διαφορετικές τιμες στα n\_in (lag observations) δίνουν διαφορετικά αποτελέσματα.Αυτό είναι εμφανές και απο τις εικόνες που προηγήθηκαν.Όσο μεγαλύτερη η τιμή τόσο χειρότερα τα αποτελέσματα.Αιτιολόγηση του γεγονότος αυτού θα επιχειρήσουμε να δώσουμε παρακάτω μαζί με τα διαγράμματα**.Πέρα απο τα διαγράμματα όμως στην περίπτωση των RNN μείωση της αποδοτικότητας δίνοντας μεγάλες τιμές n\_in οφείλεται και στο vanishing/ Exploding Gradient Problem, όπου αν η βαθμίδα (Gradient) γίνεται όλο και μικρότερη στα steps μας (δηλαδη απο t, t + 1 εως t + n ) τότε δεν μπορούμε να βρούμε συσχέτιση μεταξύ των δεδομένων του βήματος t , t + n.Τέλος δεν μπορούμε να καταλάβουμε αν έχουμε τις λάθος παραμέτρους ώστε να βρούμε αν υπάρχει στ’ αλήθεια συσχέτιση μεταξύ των δεδομένων μας στα βήματα t , t + n.**Το γεγονός αυτό είναι και ένα απο τα μειονεκτήματα των RNN.

## Ερώτημα 2 Διαγράμματα Δεδομένων

Κατα την εκτέλεση της συνάρτησης preprocess δίνοντας true στην μεταβλητή show\_plots παράγονται πληθώρα διαγραμμάτων την επεξήγηση και παρουσίαση θα κάνουμε στο κομμάτι αυτό.

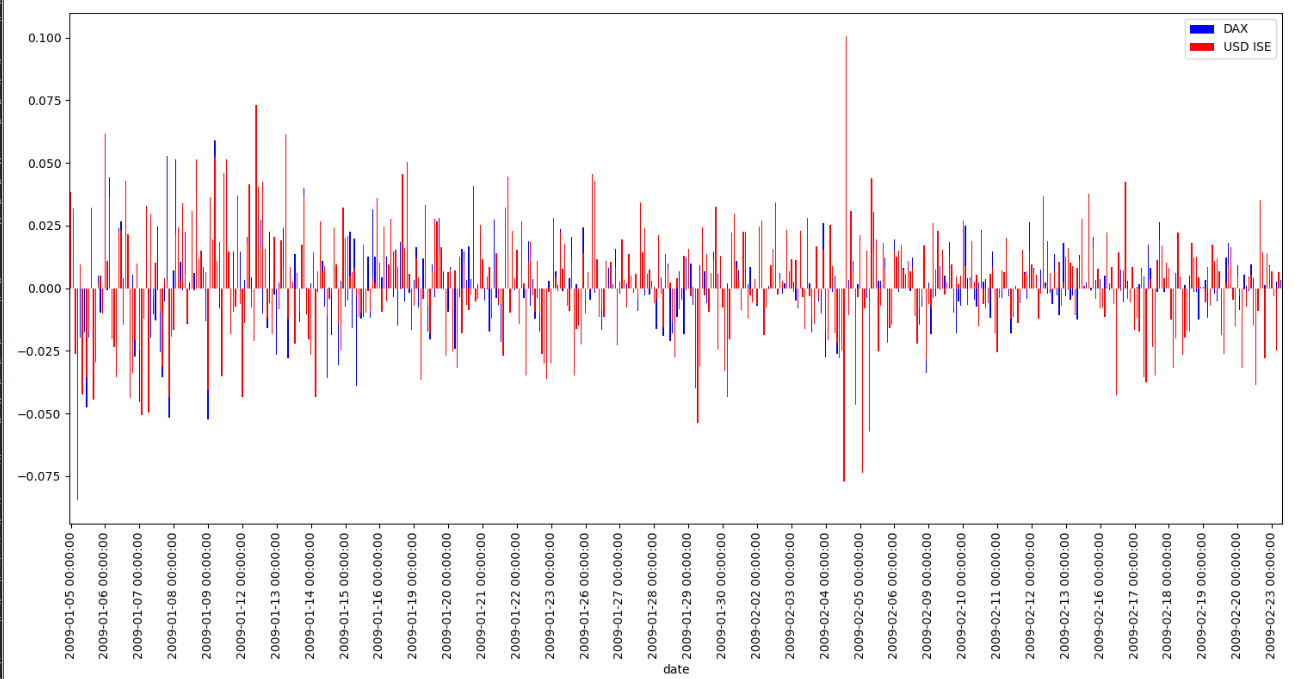
### Bar Plots

#### Bar plot SP – USD ISE



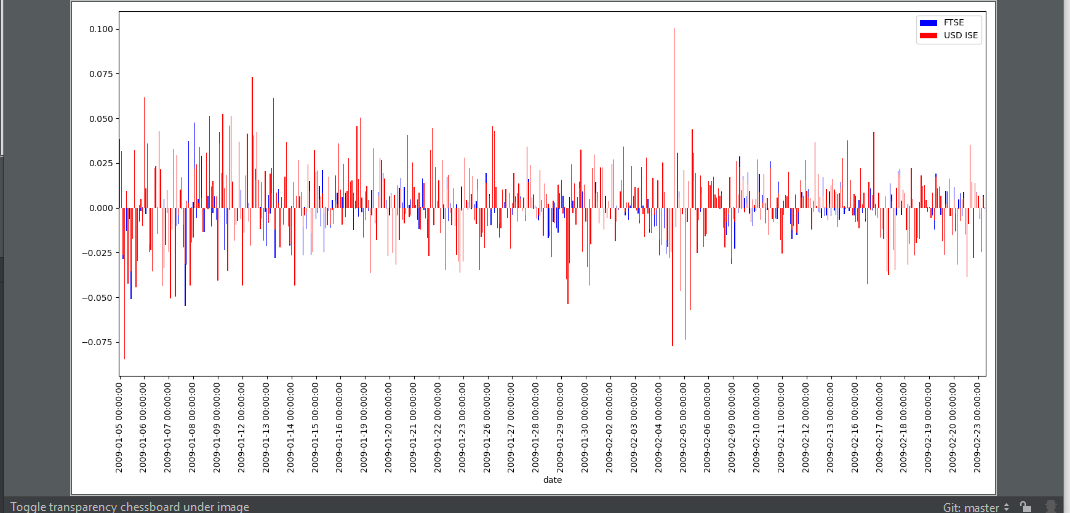
Στο συγκεκριμένο plot μπορούμε να παρατηρήσουμε σε αρκετά σημεία οτι για ίδιες ημερομηνίες κινήσεις στο δείκτη SP φαίνεται να έχουν σχέση με κινήσεις στον δείκτη USD ISE. Πιο συγκεκριμένα η άνοδος ή η κάθοδος του δείκτη SP επηρεάζει ανάλογα τον δείκτη USD ISE.

#### Bar plot DAX – USD ISE

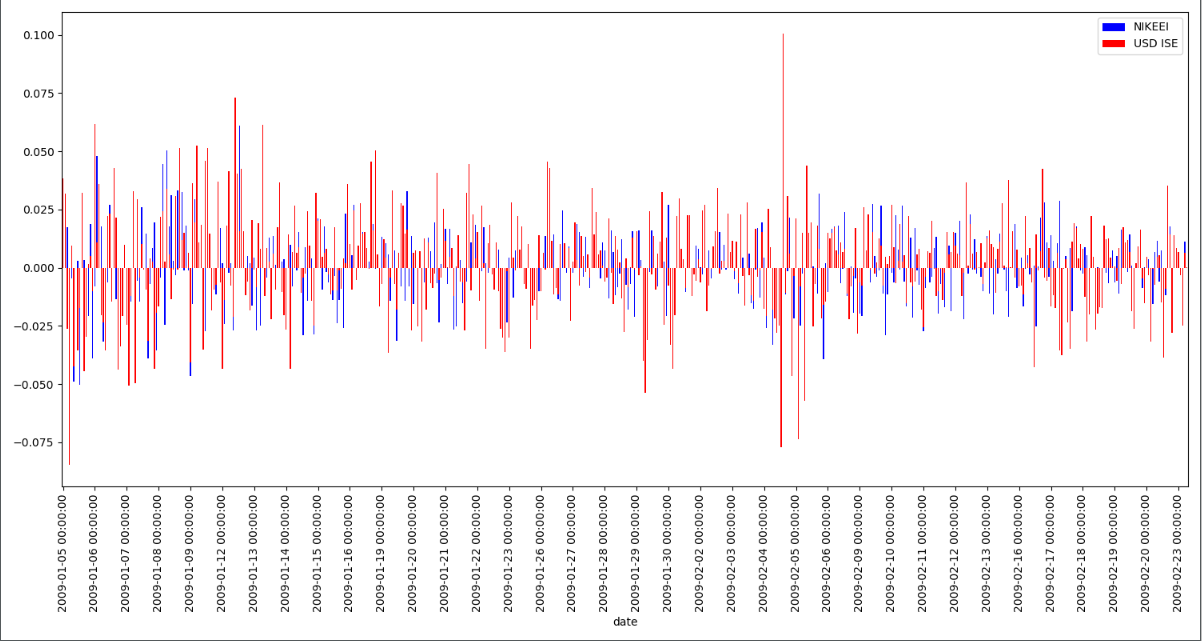


Ανάλογα αποτελέσματα μπορούμε να δούμε και σε αυτό το διάγραμμα σε αρκετά σημεία όταν οι τιμές του Dax αλλάζουν επηρεάζονται και οι τιμές του USD ISE και μάλιστα σε αρκετά σημεία προς την αντίοθετη κατεύθυνση.

#### Bar Plot FTSE – USD ISE

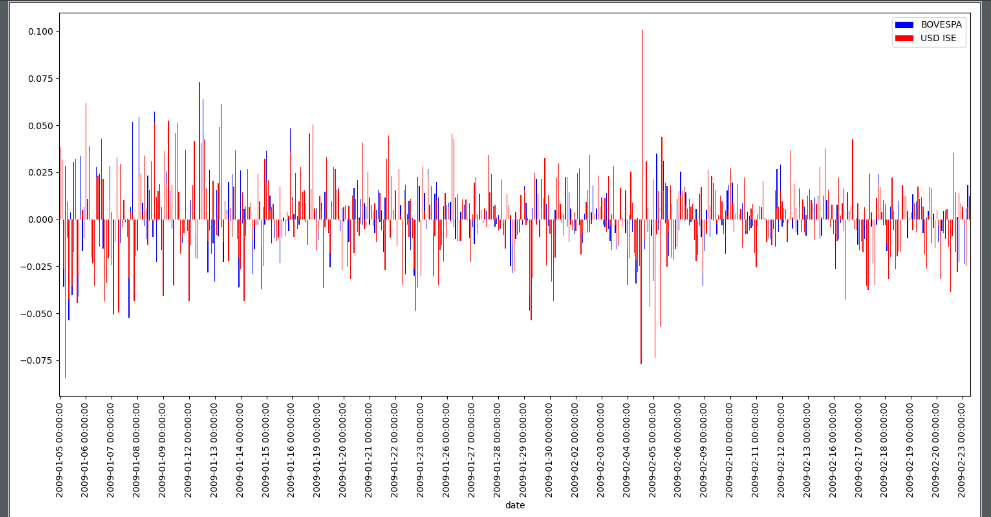


#### Bar Plot NIKKEI – USD ISE

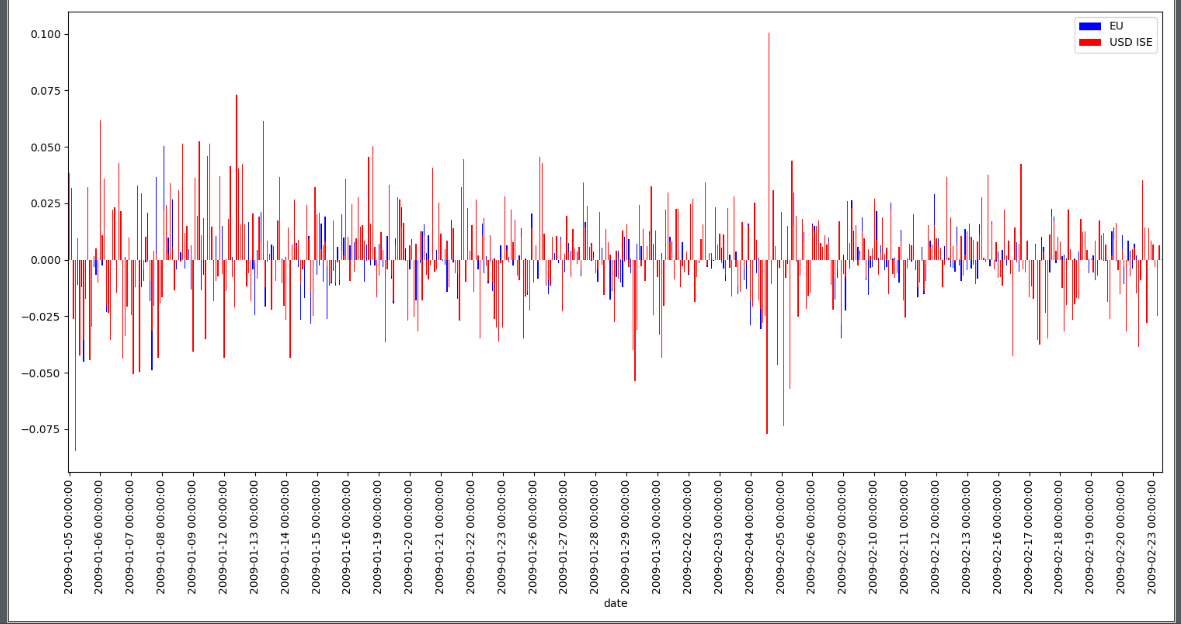


Σε πολλά σημεία ο NIKKEI με τον ISE έχουν μαζί άνοδο ή κάθοδο.

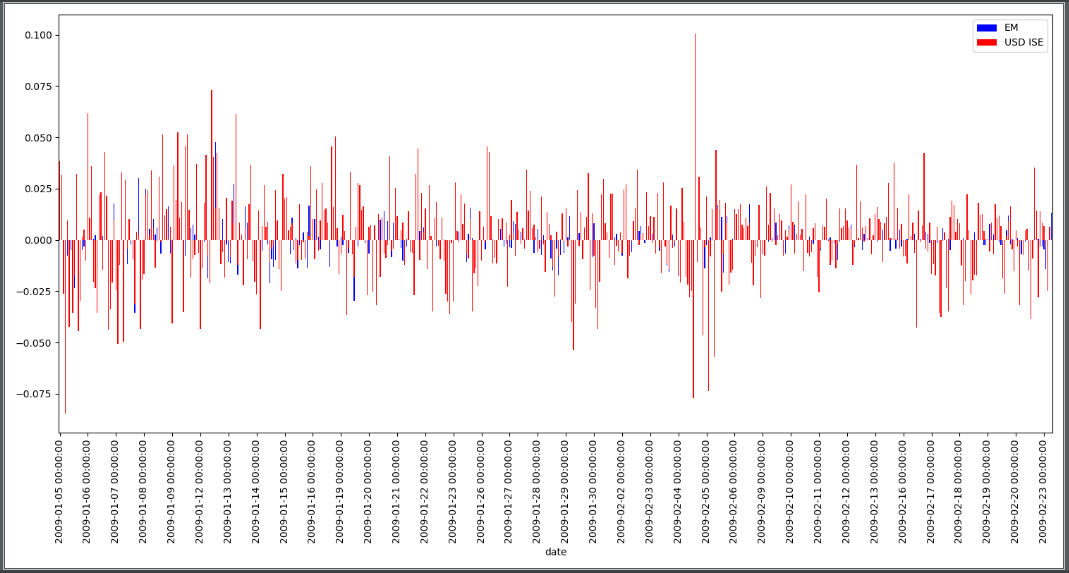
#### Bar Plot BOVESPA – USD ISE



#### Bar Plot EU – USD ISE



#### Bar Plot EM – USD ISE



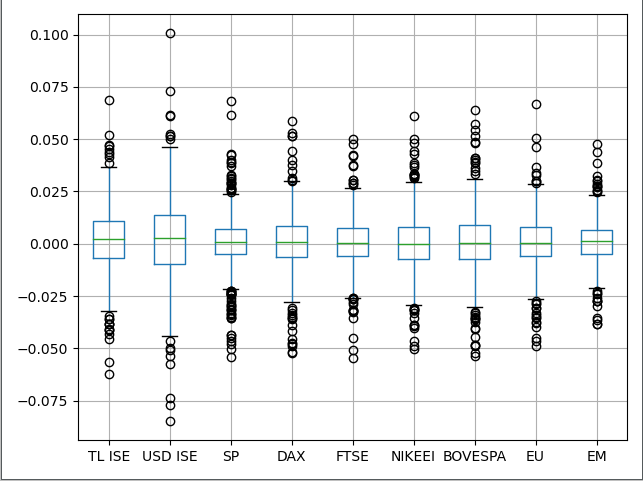
#### Συμπέρασμα

Παρατηρώντας όλα τα bar plots μπορούμε να δούμε ότι υπάρχει κάποια σχέση μεταξύ επιρροής μεταξύ των δεικτών και σε συνδιασμό με το γεγονός οτι οι δείκτες αυτοί βάση του paper : Paper: Akbilgic, O., Bozdogan, H., Balaban, M.E., (2013) A novel Hybrid RBF Neural Networks model as a forecaster, Statistics and Computing. DOI 10.1007/s11222-013-9375-7

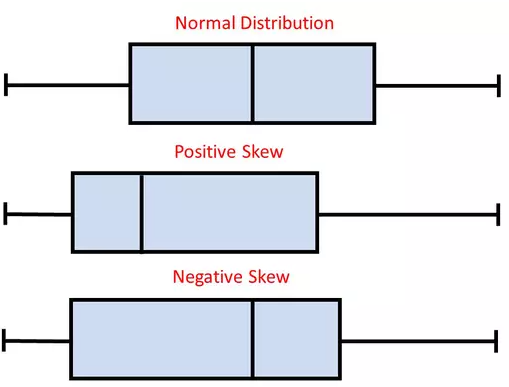
Έχουν χρησιμοποιηθεί στο παρελθόν με διαφορετικούς συνδιασμούς για την πρόβλεψη του δείκτη ISE μας ώθησαν στον να χρησιμοποιήσουμε τις τιμές τους ως features στο vector Χ.

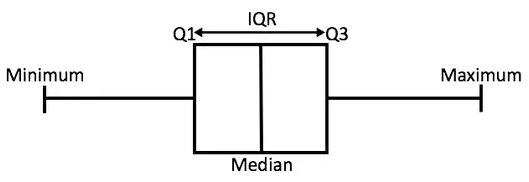
### Plots Σχετικά με το Central Tendency, Variation , Spread.

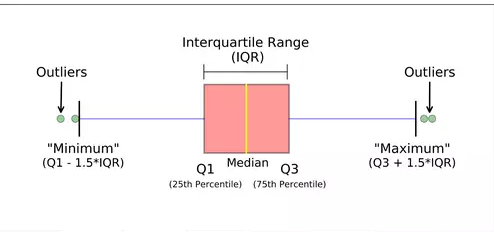
#### Box Plot



Ισχύουν τα παρακάτω:



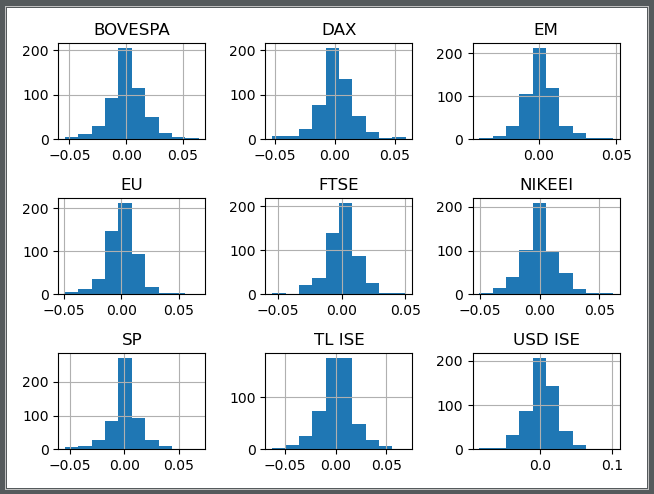




Παρατηρώντας τα Box plots βλέπουμε ότι καμία γραμμή των medians του κάθε δείκτη δεν ξεπερνάει κάποιο box plot κάποιου άλλου δείκτη οπότε είναι πολυ πιθανό οι τιμές των δεικτών μας να έχουν πολλές ομοιότητες. Επίσης το μέγεθος των κουτιών είναι σχεδόν ίδιο παντού οπότε τα δεδομένα σε γενικές γραμμές δεν είναι διασκορπισμένα μεταξύ τους.

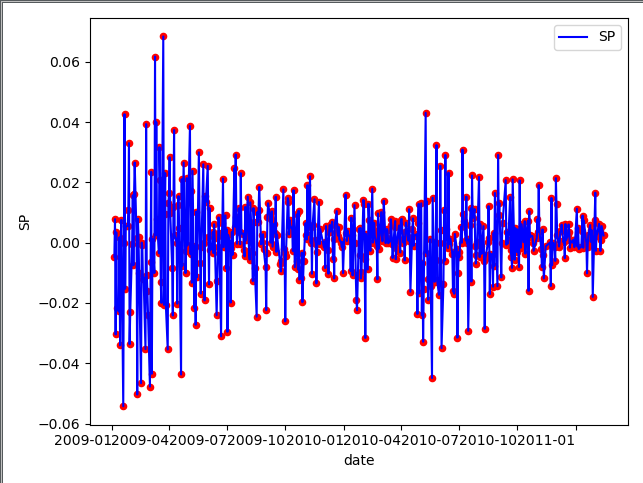
Ακόμη παρατηρούμε ότι οι τιμές που είναι τελείως εκτος των κουτιών (outliers) σε πολλά σημεία είναι όμοιες πολλές φορές η συγκεντρωσή τους ομως διαφέρει. (ΠΧ SP – FTSE). Τελος μπορούμε να δούμε οτι σε γενικές γραμμές τα δεδομένα μας έιναι συμμετρικά. (Q2 🡪 median βρίσκεται στην μέση.) ( Στα TL ISE , USD ISE 🡪 left skewed ). ( Πηγή : <https://www.khanacademy.org/math/ap-statistics/summarizing-quantitative-data-ap/stats-box-whisker-plots/v/interpreting-box-plots> )

#### Histograms



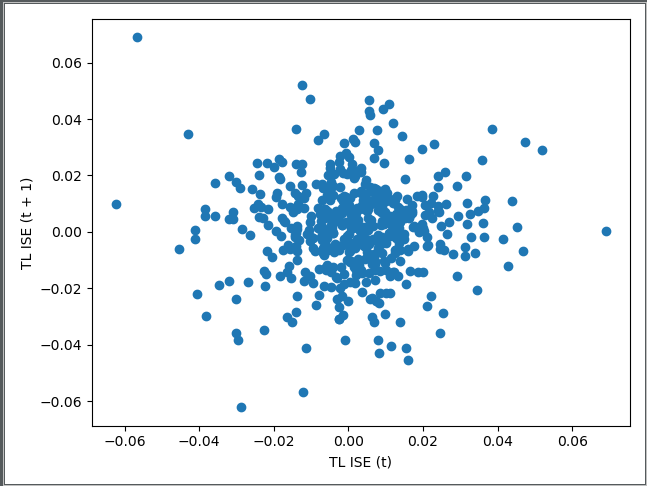
Παρατηρούμε ότι σε όλα τα ιστογράμματα των δεικτών η κατανομή των τιμών είναι σε γενικές γραμμές συμμετρική. (Πηγη: <https://www.khanacademy.org/math/probability/data-distributions-a1/displays-of-distributions/v/shapes-of-distributions> )

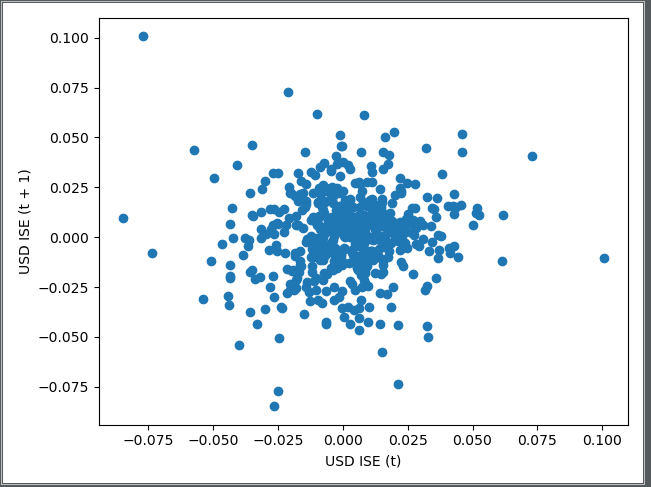
### Plot μεταξύ Date – USD Based ISE



### Lag Scatter Plots

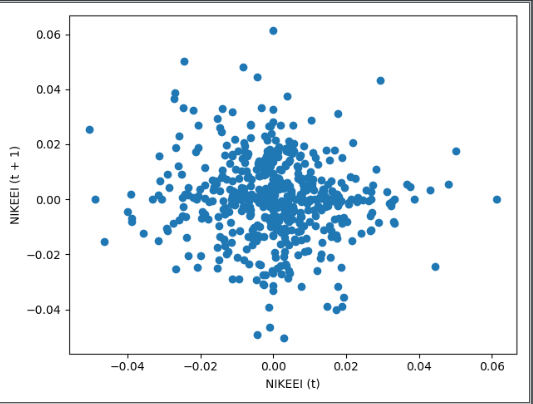
Τα Plots που ακολουθούν είναι παρόμοια συνεπώς δεν θα δώσουμε κάποιο μεμονωμένο συμπέρασμα για καθένα ξεχωριστά αλλα θα δώσουμε ένα συνολικό συμπέρασμα.

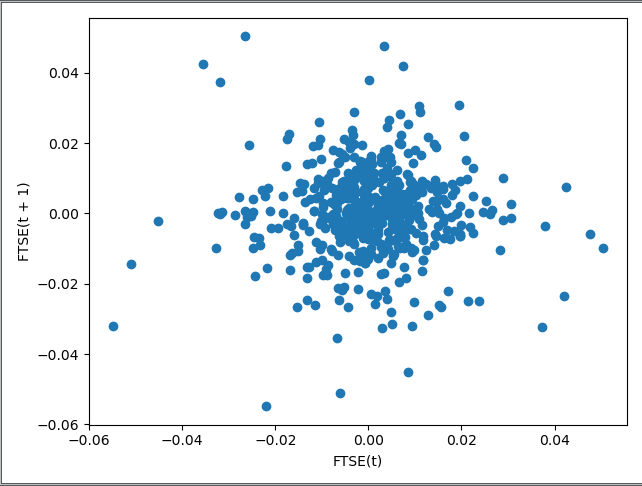


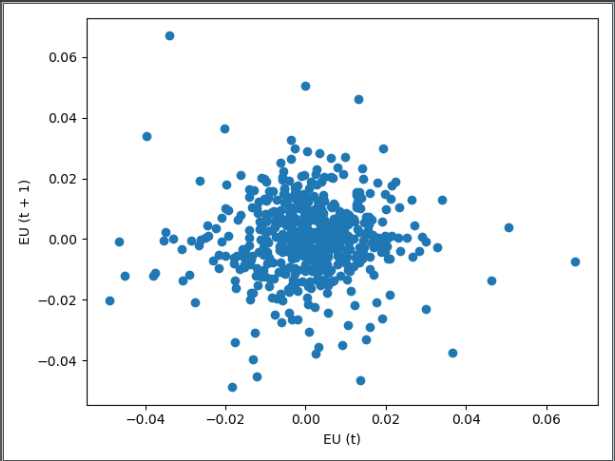


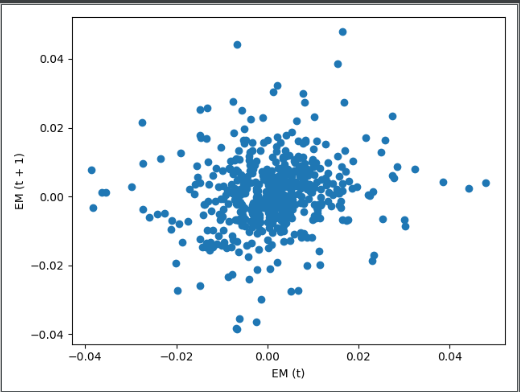


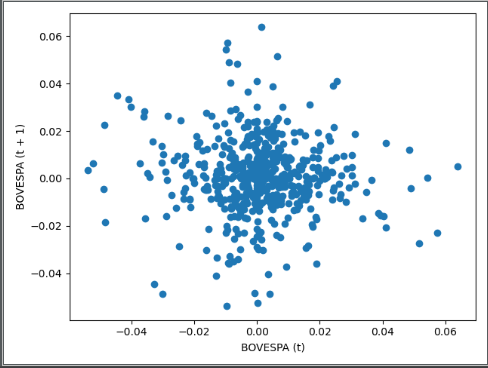








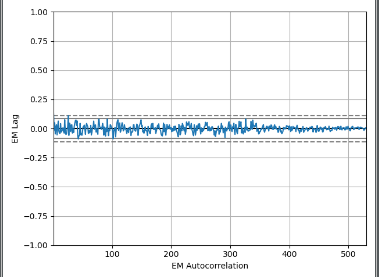




### Autocorrelation Plots

|  |  |
| --- | --- |
|  |  |

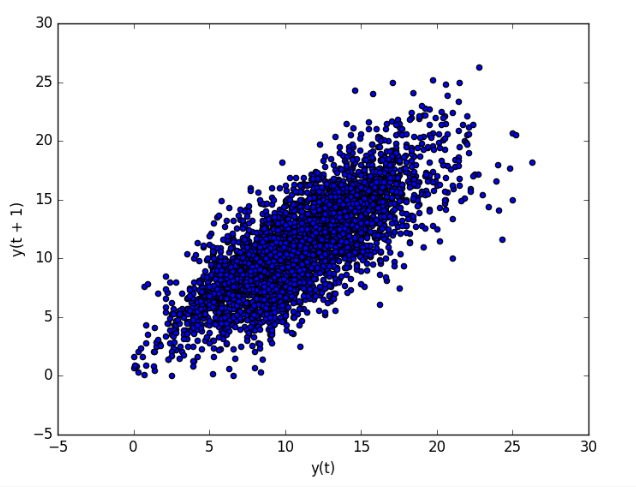
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |



Από τα παραπάνω autocorrelation Plots οι τιμές μας είναι κοντά στο μηδέν γεγονός που φανερώνει αδύναμη σχέση μεταξύ των lag observations των και δεδομένων και των πραγματικών τιμών τους.

#### Συμπέρασμα

Παρατηρώντας τα lag scatter plots και τα autocorrelation plots μπορούμε να δούμε ότι όλα τους είναι όμοια. Το πιο αξιοσημείωτο χαρακτηριστικό της ομοιότητας αυτής είναι το γεγονός οτι σε όλα τα plots παρατηρείται μεγάλη συγκέντρωση σημείων στο κέντρο με πολύ λίγες τιμές να απέχουν απο αυτό και να είναι διάσπαρτες στον χώρο.(Για τα autocorrelation plots έχουμε τιμές κοντά στο 0) Το σχήμα που μπορεί κανείς να διακρίνει από την συγκέντρωση των σημείων στο κέντρο, είναι σφαιρικό . Το γεγονός αυτο υποδεικνύει ότι τα lag observations έχουν αδύναμη έως και καθόλου σχέση. ( Αν τα σημεία συγκεντρώνοταν σε μια διαγώνια γραμμή απο τα αριστερά προς τα δεξιά η το αντίθετο τότε θα υπήρχε σχέση μεταξύ τους η οποία να μπορεί να χρησιμοποιηθεί στο μοντέλο. ) Π.Χ.



**Από την στιγμή όμως που δεν συμβαίνει κάτι τέτοιο ωθούμαστε στο να μην χρησιμοποιήσουμε τα lag observations του δείκτη USD BASED ISE στο vector Χ που έχουμε τα feature μας.**Επίσης το γεγονός ότι τα lag scatter plots μας έχουν αυτή την μορφή εξηγεί τον λόγο για τον οποίο η αύξηση των lag observations στα μοντέλα μας (n\_in) προκαλεί χαμηλότερα score. **Από την στιγμή που η σχέση μεταξύ των lag observations με τα δεδομένα μας είναι αδύναμη ή ανύπαρκτη το να τα εντάξουμε ως features για την εκπαίδευση του μοντέλου δημιουργεί παραπάνω θόρυβο με αποτέλεσμα να έχουμε πιο «φτωχές» αποδόσεις. Συνεπώς είναι λογικό το γεγονός της αύξησης των score μας όταν δοκιμάσαμε όλο και χαμηλότερες τιμες στην μεταβλητή n\_in ( lag observations ).**Ακόμη παρατηρούμε ότι δεν υπάρχει κάποια εποχηκότητα (seasonality) στα δεδομένα μας βάση σχημάτων καθώς τα δεδομένα δεν επαναλαμβάνονται με κάποια περίοδο.(Αναφερόμαστε στις τιμές και τα lag observations τους).**Τέλος στα χαμηλά score του μοντέλου μας συντέλεσε και το μικρό μέγεθος του dataset το οποίο δεν ήταν κατάλληλο για την εκπαίδευση ενός καλού μοντέλου**. Πηγές: <https://www.statisticshowto.com/lag-plot/> , <https://machinelearningmastery.com/time-series-data-visualization-with-python/>

**ΣΗΜΕΙΩΣΗ: Στην περίπτωση των RNN η μείωση της αποδοτικότητας με μεγάλες τιμές των n\_in μπορεί να οφείλεται και στο Vanishing/Exploding Gradient Decent Problem, που επεξηγήσαμε στο κομμάτι της εκτέλεσης του ερωτήματος 2 b).**

Τέλος παρατηρώντας και τα υπόλοιπα διαγράμματα βλέπουμε ότι οι τιμές των δεδομένων μας δεν είναι πολύ διάσπαρτες στον χώρο. Συνεπώς επιλέξαμε τα training set και test set να έχουν περίπου το ίδιο μέγεθος ώστε το νευρωνικό μας δίκτυο να μην εξειδικευτεί στα δεδομένα μας με αποτέλεσμα να έχουμε overfit.

## Τελικά Αποτελέσματα και Συγκρίσεις

Συγκρίνοντας τα αποτελέσματα που πήραμε απο τα υποερωτήματα 2 a) , 2 b) βλέπουμε ότι τα RNN μας έδωσαν καλύτερα score όπως και προβλέψεις.Παρόλαυτα η μείωση των score κατα την αύξηση των lag observations (n\_in (steps) ) επηρέαζε σημαντικότερα τα RNN σε σχέση με τα MLP.Όπως έχει αναφερθεί και πιο πριν το μικρό μέγεθος του dataset και η κατανομή των τιμών του δεν βοήθησαν στο να κάνουμε καλά μοντέλα.

|  |  |
| --- | --- |
| RNN | MLP |
|  |  |
|  |  |

## Πηγές

* Paper: Akbilgic, O., Bozdogan, H., Balaban, M.E., (2013) A novel Hybrid RBF Neural Networks model as a forecaster, Statistics and Computing. DOI 10.1007/s11222-013-9375-7
* <https://machinelearningmastery.com/>
* <https://www.statisticshowto.com/>
* <https://www.khanacademy.org/>
* <https://www.dummies.com/>
* Σημειώσεις εργαστηρίων python
* Σημειώσεις προεπεξεργασίας δεδομένων Dateprep1/2
* Keras Docs: <https://faroit.com/keras-docs/1.2.0/>
* Pandas Docs: <https://pandas.pydata.org/docs/>
* Matplotlib Docs: <https://matplotlib.org/3.2.1/contents.html>