ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ ΚΑΙ ΑΣΦΑΛΙΣΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ»

ΣΤΑΤΙΣΤΙΚΕΣ ΜΕΘΟΔΟΙ ΣΤΗΝ ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ

ΜΠΟΥΡΑΝΤΑΝΗΣ ΑΝΔΡΕΑΣ

mpouras.math@gmail.com

ΑΜ:ΜΕΣ22003

Στην παρούσα εργασία θα χρησιμοποιήσουμε μεθόδους εξόρυξης δεδομένων και τεχνικές μηχανικής μάθησης στο σύνολο δεδομένων Cardiotocography Data Set από το UCI ML Repository.

https://archive.ics.uci.edu/ml/datasets/Cardiotocography

Το σύνολο δεδομένων Cardiotocography αποτελείται από 2126 καρδιοτογράμματα (CTGs), από τα οποία μετρήθηκαν 21 χαρακτηριστικά. Τα CTGs ταξινομήθηκαν από τρεις ειδικούς σε σχέση με το μορφολογικό μοτίβο FHR (10 κατηγορίες) και την κατάσταση του εμβρύου NSP (3 κατηγορίες).

1 .) Preprocessing

Θα ξεκινήσουμε κάνοντας επιλογή των δεδομένων και θα προχωρήσουμε κάνοντας ορισμένες προπαρασκευαστικές εργασίες.

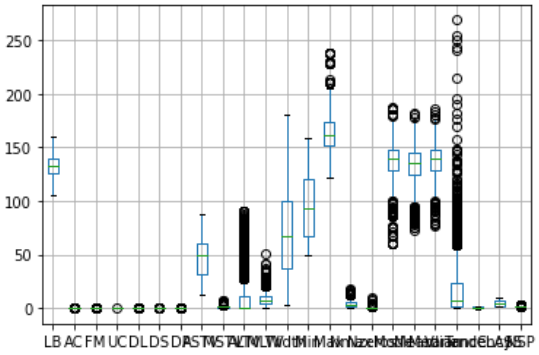
	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 36	Unnamed: 37	Unnamed: 38	Unn
0	b	e	AC	FM	UC	DL	DS	DP	DR	NaN	...	E	AD	DE	
1	240	357	0	0	0	0	0	0	0	NaN	...	-1	-1	-1	
2	5	632	4	0	4	2	0	0	0	NaN	...	-1	1	-1	
3	177	779	2	0	5	2	0	0	0	NaN	...	-1	1	-1	
4	411	1192	2	0	6	2	0	0	0	NaN	...	-1	1	-1	

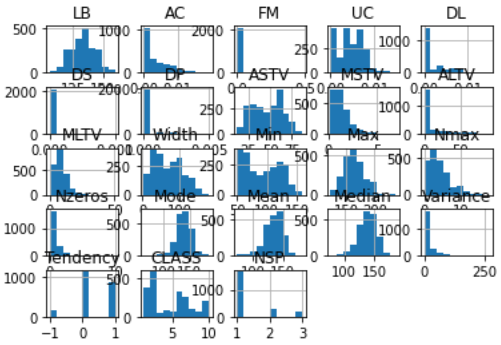
5 rows × 46 columns

Από τις 46 στήλες των δεδομένων και τις 2130 γραμμές θα επιλέξουμε τις 23 στήλες και 2126 γραμμές που θέλουμε για την ανάλυση.

	LB	AC	FM	UC	DL	DS	DP	ASTV	MSTV	ALTV	...	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency	CLASS	NSP
1	120	0	0	0	0	0	0	73	0.5	43	...	126	2	0	120	137	121	73	1	9	2
2	132	0.00638	0	0.00638	0.00319	0	0	17	2.1	0	...	198	6	1	141	136	140	12	0	6	1
3	133	0.003322	0	0.008306	0.003322	0	0	16	2.1	0	...	198	5	1	141	135	138	13	0	6	1
4	134	0.002561	0	0.007682	0.002561	0	0	16	2.4	0	...	170	11	0	137	134	137	13	1	6	1
5	132	0.006515	0	0.008143	0	0	0	16	2.4	0	...	170	9	0	137	136	138	11	1	2	1

Επιπλέον θα δούμε κάποια plots για να αποκτήσουμε κάποια διορατικότητα για τα features:





Αρχικά βλέπουμε αρκετά outliers στο boxplot ωστόσο μπορεί να περιέχουν χρήσιμη πληροφορία στην περεταίρω ανάλυση. Για το λόγο αυτό θα δημιουργήσουμε ένα νέο dataset όπου θα χειριστούμε τα outliers ενώ θα διατηρήσουμε και το αρχικό.

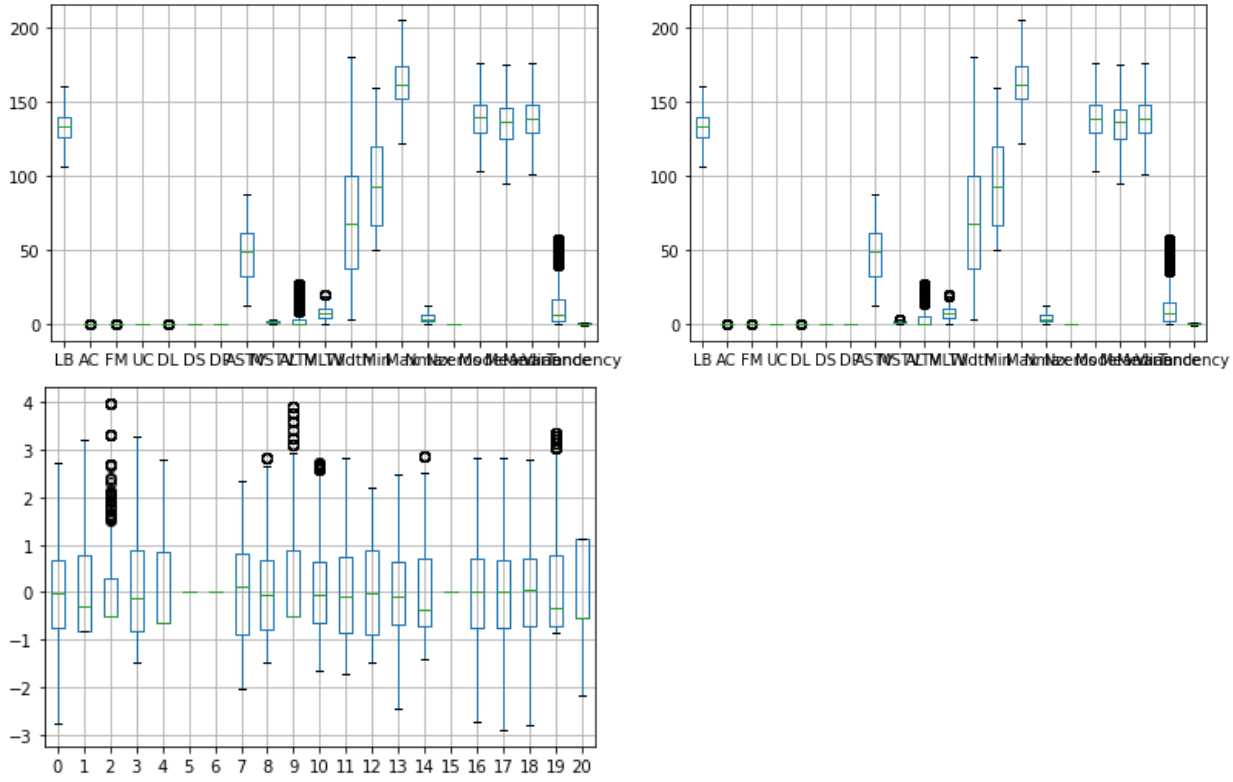
Επειδή το δείγμα αποτελείται από ολίγες παρατηρήσεις δεν έχουμε την πολυτέλεια απλά να αφαιρέσουμε τα outliers. Επομένως η διαδικασία με την οποία θα διαχειριστούμε τα outliers είναι η εξής. Θα θεωρήσουμε τα outliers ως missing values και θα αντικαταστήσουμε την τιμή τους με τη μέση τιμή που έχουνε παρατηρήσεις που βρίσκονται στην ίδια κλάση. Ωστόσο επειδή στην προηγούμενη μέθοδο παρατηρήσεις που δεν είναι κοντά στο outlier θα επηρεάσουν την τιμή του outlier(λόγω προσέγγισης μέσω mean) ένας άλλος αλγόριθμος που μπορεί να αντιμετωπίσει αυτό το πρόβλημα είναι ο KNN όπου η τιμή του outlier θα συμπληρωθεί μέσω των 10 κοντινότερων παρατηρήσεων και με βάρη ανάλογα της απόστασης τους. Επειδή η απόσταση αποτελεί παράγοντας στην τελευταία μέθοδο θα χρειαστεί να κάνουμε scaling στα δεδομένα πριν την εφαρμόσουμε.

Εκτελώντας την διαδικασία ως προς την 'CLASS' διαχειριστήκαμε 1571 από τα 1982 outliers .

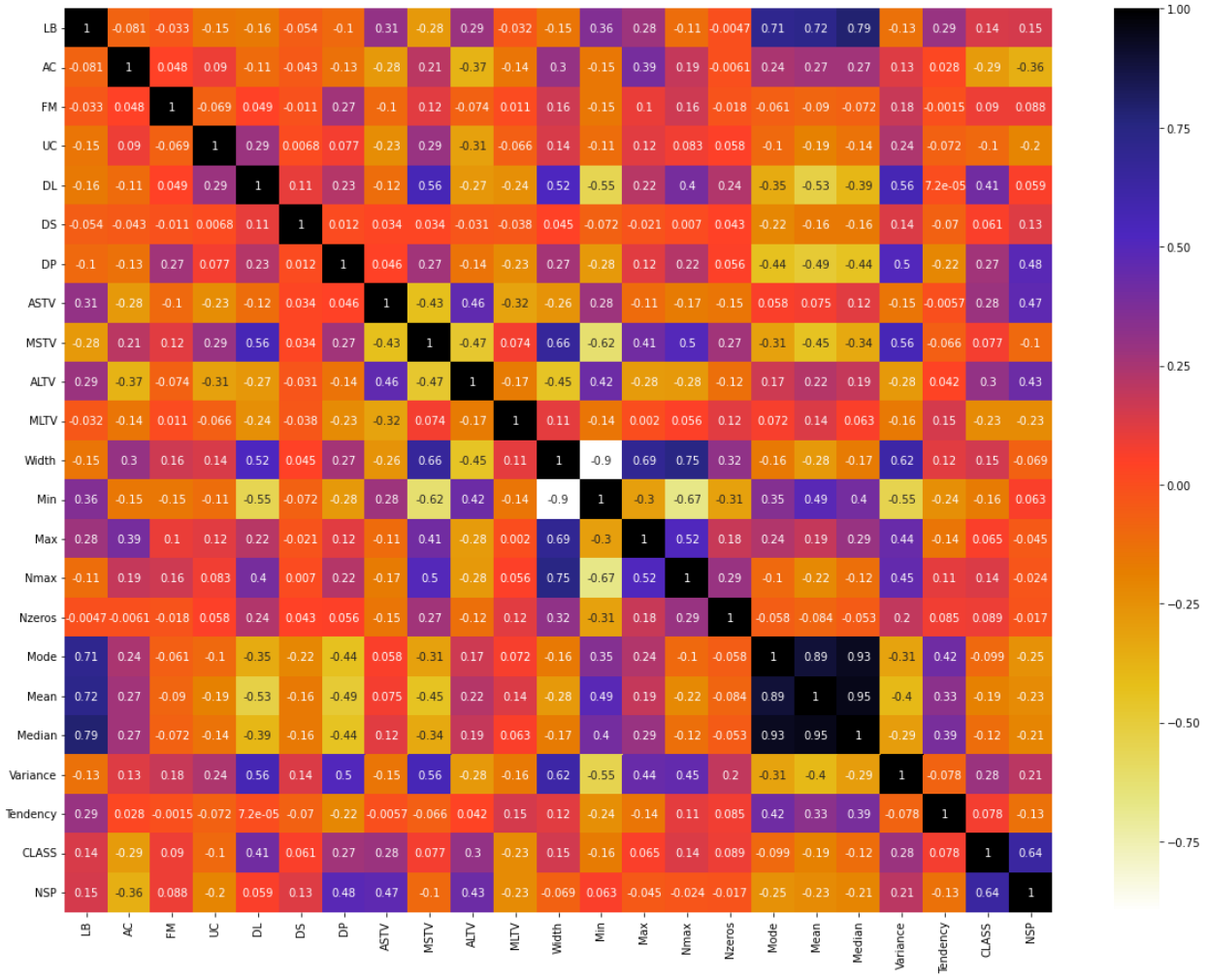
Εκτελώντας την διαδικασία ως προς την 'NSP' διαχειριστήκαμε 1145 από τα 1982 outliers.

Με τον αλγόριθμο KNN διαχειριστήκαμε 1675 από τα 1982 outliers.

Ας δούμε τα boxplot των δεδομένων μετά από τον κάθε αλγόριθμο αντίστοιχα:



(!Με τα data του KNN παρατηρήθηκε overfitting στο classification οπότε θα κρατήσουμε το mean by group(CLASS) data για το classification και το KNN για το clustering.)

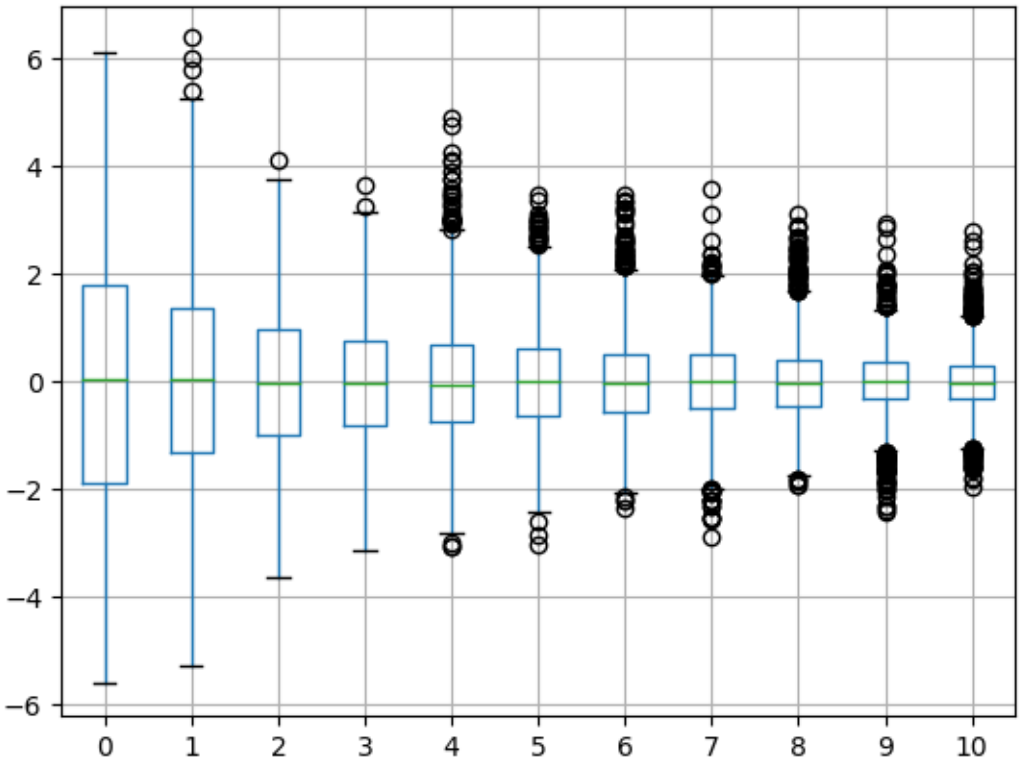


Καθώς βλέπουμε αρκετά features να είναι συσχετισμένα μεταξύ τους θα επιδιώξουμε να μειώσουμε τις διαστάσεις των δεδομένων με Ανάλυση σε Κύριες ΣυνιστώσεςPCA) δημιουργώντας νέες ασυσχέτιστες μεταβλητές με φθίνουσα μεταβλητότητα(η πρώτη θα έχει το μεγαλύτερο ποσοστό πληροφορίας από τα δεδομένα ενώ η τελευταία το μικρότερο).

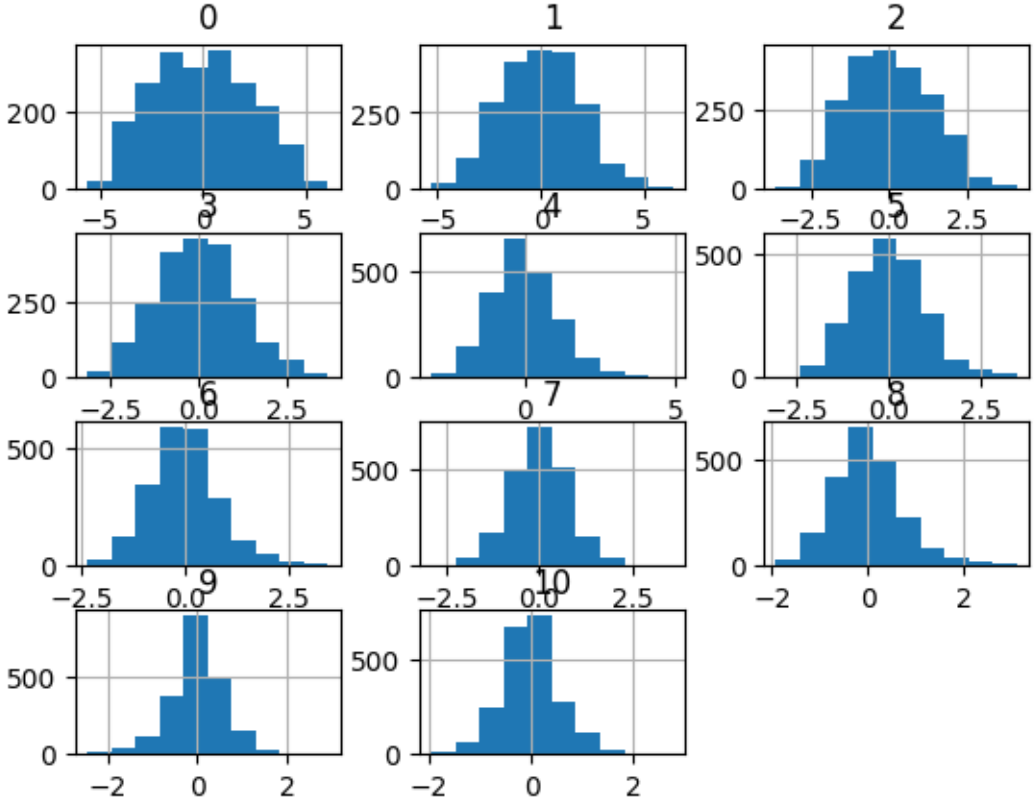
```
[0.3197, 0.2009, 0.0949, 0.0782, 0.0671, 0.0516, 0.0408, 0.0335,
 0.0288, 0.0209, 0.0179, 0.0145, 0.0114, 0.0071, 0.0067, 0.0039,
 0.0014, 0.0006, 0.    , 0.    , 0.    ])
```

Στον παραπάνω πίνακα αναγράφεται η αντίστοιχη διασπορά της κάθε μεταβλητής μετά την PCA.Δηλαδή με την πρώτη συνιστώσα κρατάμε το 31.97% της μεταβλητότητας(πληροφορίας) των δεδομένων,με τις πρώτες δύο το 52.06% κ.ο.κ.

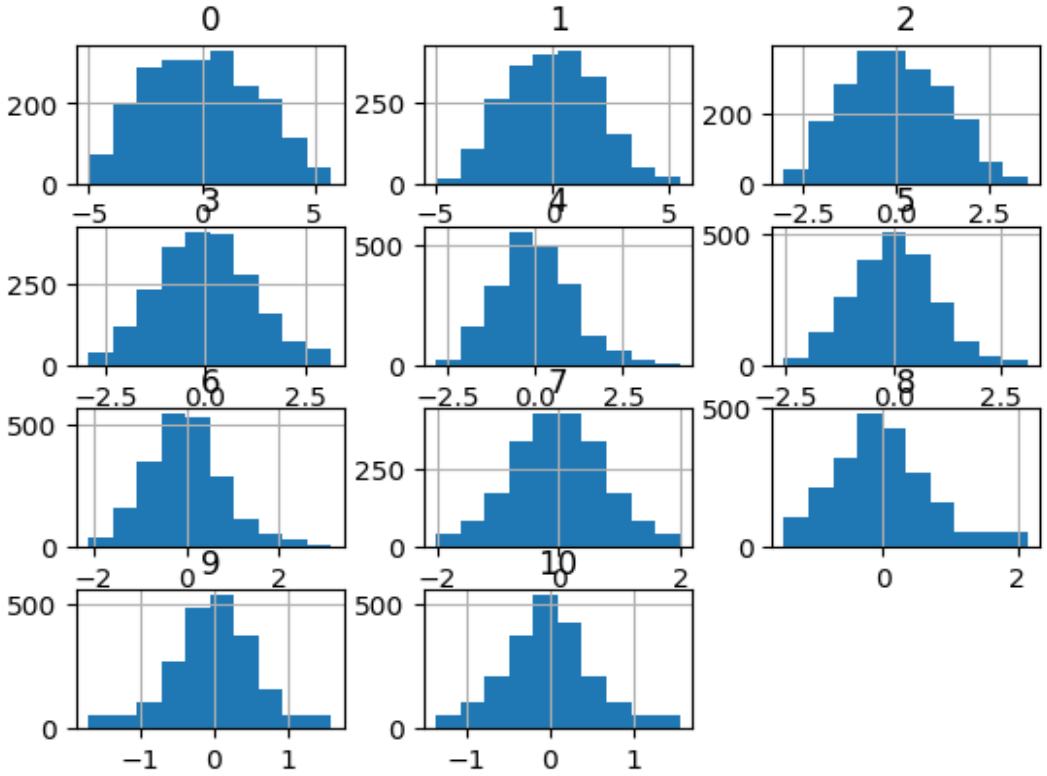
Με τις 11 πρώτες συνιστώσες διατηρούμε το 95.4% της αρχικής μεταβλητότητας συνεπώς θα αφαιρέσουμε τις υπόλοιπες 10 από τα δεδομένα.

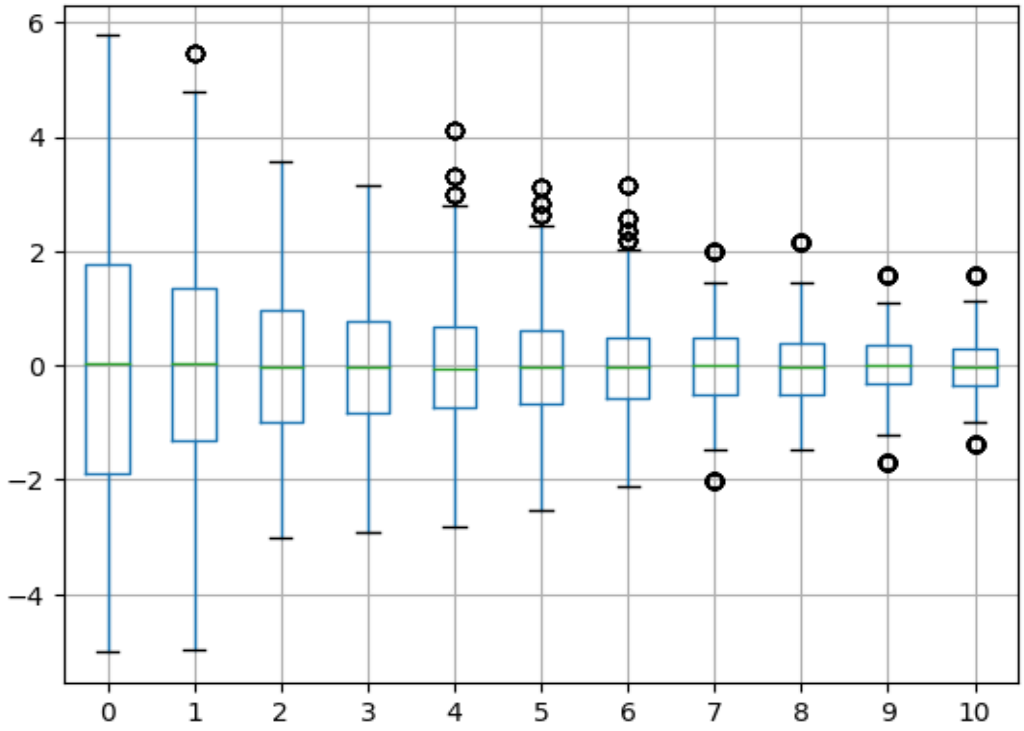


Επιπλέον αν θέλουμε να ανταλλάξουμε λίγη ακρίβεια για numerosity reduction μπορούμε να διακριτοποιήσουμε τα δεδομένα μας με τη μέθοδο Bin by group mean(CLASS).



Κάνοντας την διακριτοποίηση θα δούμε τις αλλαγές μέσω διαγραμμάτων.



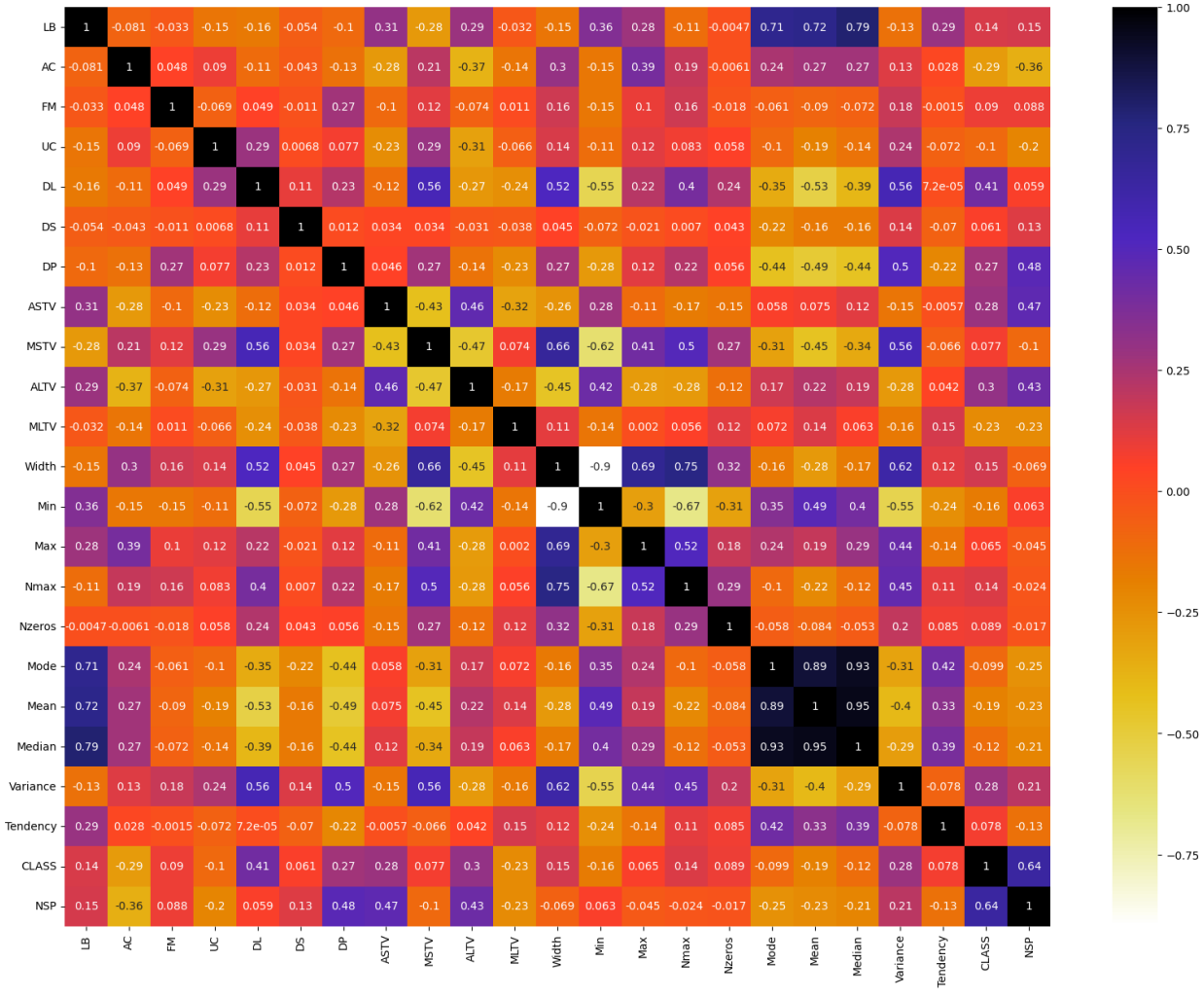


Βλέπουμε πως στα ιστογράμματα δεν υπήρξαν μεγάλες διαφορές ενώ στο boxplot βλέπουμε πως χειριστήκαμε και outliers στα περισσότερα features.

(!Γενικά δε θα χρησιμοποιήσουμε τα διακριτοποιημένα δεδομένα στη συνέχεια)

Classification

Θέλουμε να εκπαιδεύσουμε μία μέθοδο classification για το Cardiotocography Dataset η οποία να μαθαίνει από τα 21 χαρακτηριστικά και να κατηγοριοποιεί σε 10 μορφολογικά μοτίβα (CLASS). Πρώτο βήμα είναι να επιλέξουμε τα features που θα μεγιστοποιήσουν την απόδοση της μεθόδου. Από τη μία θέλουμε features που να είναι αρκετά συσχετισμένες με την κατηγορία αλλά παράλληλα θέλουμε να έχουν και μεγάλη διασπορά(πληροφορία). Με το heatmap μπορούμε να δούμε ποια features έχουν ισχυρή συσχέτιση με την κατηγορία καθώς και ποια χαρακτηριστικά είναι συσχετισμένα μεταξύ τους.



Παρατηρούμε πως αρκετά χαρακτηριστικά(features) έχουν μεγάλη τιμή συσχέτισης και για τον λόγο αυτό αξίζει να κάνουμε Ανάλυση Κύριων Συνιστωσών(PCA) στα δεδομένα καθώς οι καινούριες μεταβλητές θα είναι ασυσχέτιστες μεταξύ τους. Οπότε θα εκπαιδεύσουμε έναν αλγόριθμο στο αρχικό dataset αλλά και σε ένα που έχει προηγηθεί PCA και θα συγκρίνουμε τα αποτελέσματα.Ο αλγόριθμος classification που θα χρησιμοποιήσουμε είναι ο **Support Vector Machines**(classification).

Κανονικό Dataset (με outliers)

Θα ξεκινήσουμε κάνοντας feature selection.Θέλουμε γενικά features που να είναι συσχετισμένα με την μεταβλητή-στόχο(‘CLASS’) οπότε αν θέλαμε απλά να πάρουμε τις 2 πιο συσχετισμένες με την ‘CLASS’ τότε από τον πίνακα θα διαλέγαμε την 4η και 19η και θα προέκυπταν τα εξής αποτελέσματα.

```
Prediction accuracy for the training dataset is:  55.6 %
Prediction accuracy for the testing dataset is:  55.4 %
Prediction balanced accuracy for the training dataset is:  33.5 %
Prediction balanced accuracy for the testing dataset is:  34.9 %
```

Ωστόσο εκτός της συσχέτισης θέλουμε και features που να έχουν μεγάλη διασπορά (πληροφορία). Θα επιτύχουμε τον στόχο αυτό με το library mutual_info_classif της scikit learn. Αν θέλουμε να χρησιμοποιήσουμε 2 μεταβλητές το mutual_info_classif μας προτείνει τις εξής 2:

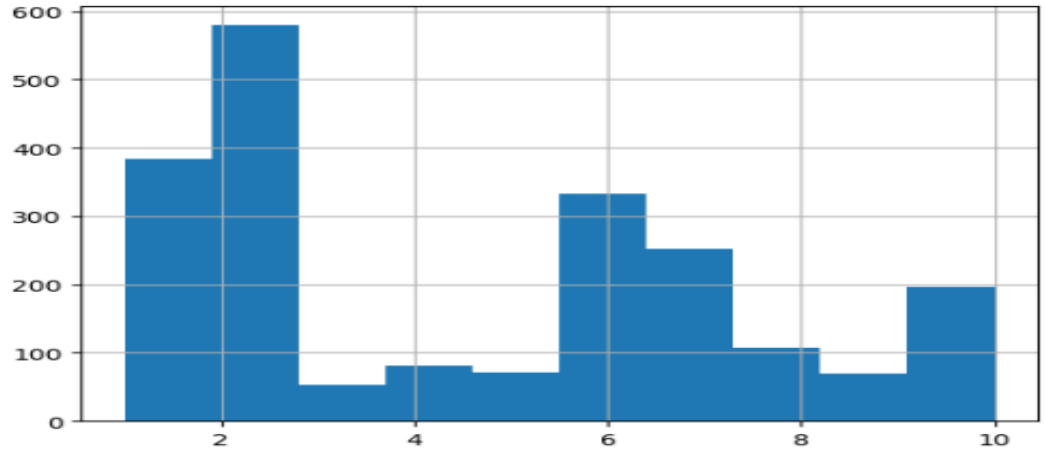
```
selector=SelectKBest(mutual_info_classif,k=2)
selector.fit(X,Y)
selector.transform(X)
X.columns[selector.get_support()]
```

```
Int64Index([1, 19], dtype='int64')
Prediction accuracy for the training dataset is:  64.4 %
Prediction accuracy for the testing dataset is:  62.9 %
Prediction balanced accuracy for the training dataset is:  39.6 %
Prediction balanced accuracy for the testing dataset is:  39.8 %
```

Οπότε βλέπουμε ότι επιτυγχάνεται μεγαλύτερη ακρίβεια με τα τελευταία features.
Αν και 62.9% αποτελεί καλός αριθμός δεδομένου ότι συμπεριλήφθηκαν μόνο δύο μεταβλητές θα επιδιώξουμε ένα καλύτερο ποσοστό προσθέτοντας επιπλέον features.Επιπρόσθετα παρατηρούμε πως ο αλγόριθμος έχει πρόβλημα στο να κατηγοριοποιεί στοιχεία που προέρχονται από την κλάση 5.

	precision	recall	f1-score	support
class 1	0.6042	0.7632	0.6744	76
class 2	0.8491	0.8738	0.8612	103
class 3	0.0000	0.0000	0.0000	10
class 4	1.0000	0.6667	0.8000	12
class 5	0.8000	0.2353	0.3636	17
class 6	0.8219	0.9231	0.8696	65
class 7	0.9661	0.9194	0.9421	62
class 8	1.0000	0.9091	0.9524	22
class 9	0.9167	0.7333	0.8148	15
class 10	0.6809	0.7273	0.7033	44

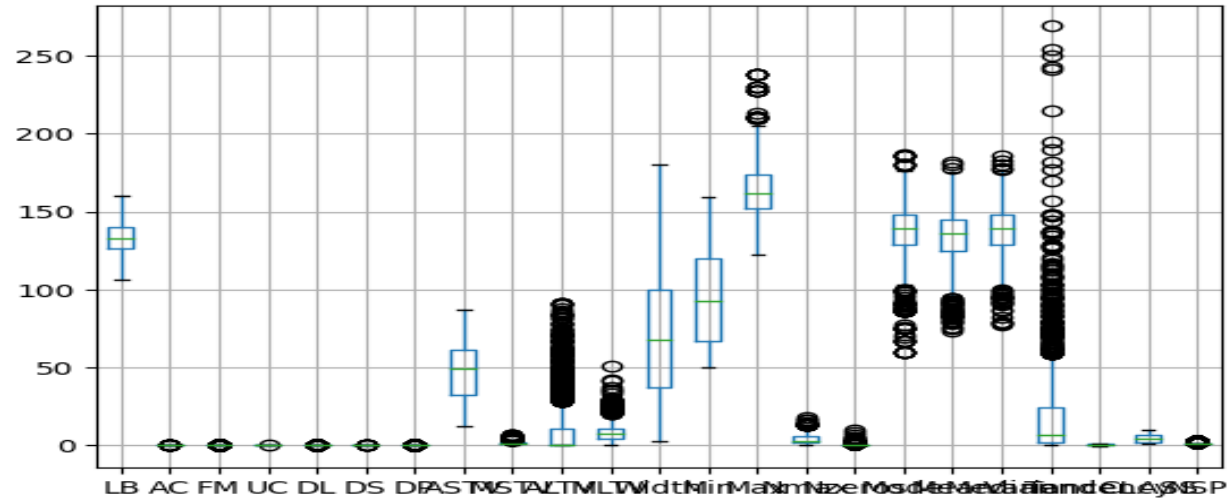
Αυτό λογικά οφείλεται στο ότι ο αλγόριθμος χρειάζεται παραπάνω πληροφορία(features) για να αναγνωρίζει στοιχεία από την συγκεκριμένη κλάση όπως επίσης ότι δεν έχουμε αρκετές παρατηρήσεις στην συγκεκριμένη κλάση ώστε να εκπαιδευτεί ο classificator.



Θα δούμε τώρα τα αποτελέσματα που προκύψαν από διαφορετικό συνδυασμό features.

Number of features	Features	Pred. Acc. Train Data (%)	Pred. Acc. Test Data (%)	Balan. Acc. Train Data (%)	Balan. Acc. Test Data (%)	Observations
2	1,19	64.4	62.9	39.6	39.8	
3	1,4,19	69.9	71.1	44.2	48	
4	1,4,8,19	75.3	77	52.2	54.4	
5	1,4,8,11,19	75.7	77.9	53.3	56.9	
6	1,4,8,11,12,19	76.1	76.3	53.3	55.3	*
6	1,4,7,8,11,19	79	79.1	59.5	61.9	
7	1,4,7,8,11,17,19	81.5	79.6	63.7	63.9	
8	1,4,7,8,10,11,17,19	82.6	82.2	68.8	72.6	
9	1, 4, 6,7, 8, 10, 11,, 17, 19	82,5	82.9	68,4	72,2	**
10	1, 4, 6,7, 8,10, 11,14, 17, 19,20	83.4	83.6	68.8	72.2	***
21	Πλήρες μοντέλο	85.2	81.5	72,4	75	

(*) Βλέπουμε ότι με την προσθήκη της 12^{ης} μεταβλητής που προτάθηκε από το mutual_info_classif η απόδοση του αλγορίθμου μειώθηκε. Επομένως θα επιλέξουμε ένα feature από το heatmap. Παρατηρούμε επίσης ότι από εδώ και πέρα κάθε προσθήκη feature δεν προσφέρει τόση ακρίβεια (1-2%).
(**) Ακόμα ο αλγόριθμος δυσκολεύεται να αναγνωρίσει στοιχεία που προέρχονται από την κλάση 5 -precision 0%.
(***) Το precision για την κλάση 5 παύει να είναι μηδέν ωστόσο παραμένει ακόμα χαμηλό 11%., δηλαδή η μεταβλητή 14 έχει κάποια παραπάνω πληροφορία για την κλάση 5. Επιπλέον βλέπουμε πως οποιαδήποτε προσθήκη feature πλέον δεν προσφέρει παραπάνω απόδοση στον αλγόριθμο.

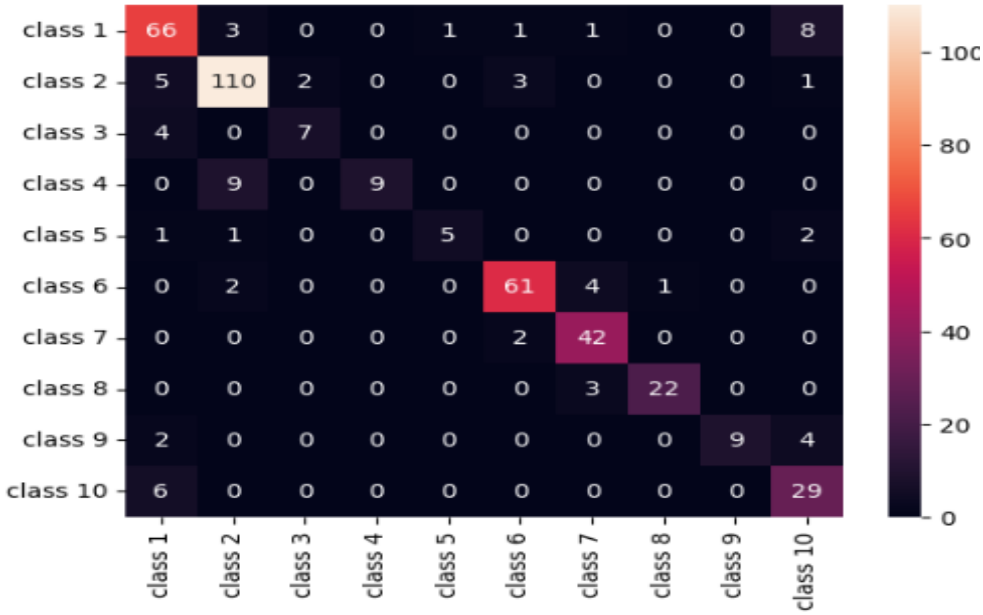


Καθώς η προηγούμενη ανάλυση έγινε χωρίς απαλοιφή των outliers θα επαναλάβουμε την διαδικασία με τα δεδομένα όπου έχουμε αντιμετωπίσει τα outliers. Στα δεδομένα όπου τα χειριστήκαμε παρατηρήθηκε overfitting επομένως θα δούμε τα αποτελέσματα με το dataset όπου τα outliers συμπληρωθήκαν με το by-class mean.

Κανονικό Dataset(χωρίς outliers)

Αφού έχουμε προσαρμόσει τα outliers βλέπουμε πως τα features DP,DS και Nzeros έχουνε πλέον σχεδόν μηδενική διασπορά επομένως μπορούμε να τα βγάλουμε καθώς δεν καλυτερεύουν την απόδοση του αλγορίθμου. Χρησιμοποιώντας τα υπόλοιπα 18 features προκύπτει το εξής report:

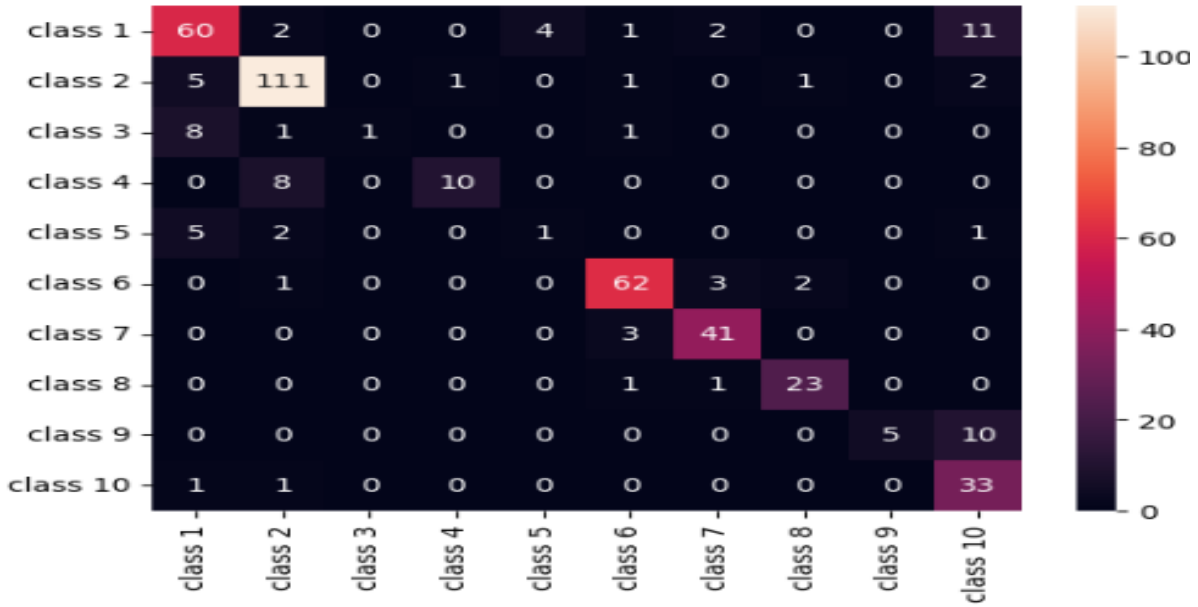
```
Prediction accuracy for the training dataset is:  88.9 %
Prediction accuracy for the testing dataset is:  84.5 %
Prediction balanced accuracy for the training dataset is:  83.5 %
Prediction balanced accuracy for the testing dataset is:  75.9 %
```



Ωστόσο παρατηρήθηκε πως ξεκινώντας από κανένα feature και σταδιακά προσθέτοντας ένα ένα τα features το pred test accuracy ανέβαινε σε κοντινές τιμές με το pred train accuracy ως ότου να φτάσει 81.5% με τον συνδυασμό των παρακάτω features:

```
selector=SelectKBest(mutual_info_classif,k=8) #το mutual_info επιμενει για την 12 ωστοσο παλι δεν κερδιζουμε ακριβεια
selector.fit(X,Y)                 #οπότε θα αυξήσουμε το k κατα μια μοναδα και θα παρουμε την αμεσως επομενη καλυτερη επιλογη
selector.transform(X)
X.columns[selector.get_support()]
```

```
Int64Index([1, 4, 7, 8, 9, 11, 12, 19], dtype='int64')
Prediction accuracy for the training dataset is:  81.7 %
Prediction accuracy for the testing dataset is:  81.5 %
Prediction balanced accuracy for the training dataset is:  67.6 %
Prediction balanced accuracy for the testing dataset is:  64.6 %
```



Ωστόσο φαίνεται στο confusion matrix πως ο αλγόριθμος δυσκολεύεται να προβλέψει σωστά στοιχεία από την κλάση 5 γεγονός που ρίχνει ελαφρώς την γενική ακρίβεια του αλλά ιδιαίτερα την ισορροπημένη ακρίβεια του.
Αν αυτό δεν αποτελεί πρόβλημα ο παραπάνω συνδυασμός το βέλτιστο feature selection καθώς επιτυγχάνεται με ακρίβεια πάνω του 80% με μόνο 8 μεταβλητές από τις 21 και με προσθήκη παραπάνω η απόδοση του δεν αυξάνεται πολύ ενώ παράλληλα σχηματίζεται overfitting issue.

Αν ωστόσο η πρόβλεψη της συγκεκριμένης κλάσης είναι σημαντική, δεδομένου ότι ακόμα και με την προσθήκη όλων των 21 features το recall για την κλάση αυτή είναι προσεγγιστικά 55%, μπορούμε οικονομικότερα να καταφέρουμε αντί αυτού 44% χρησιμοποιώντας τον συνδυασμό:

```
selector=SelectKBest(mutual_info_classif,k=11)
selector.fit(X,Y)
selector.transform(X)
X.columns[selector.get_support()]
```

```
Int64Index([1, 2, 4, 7, 8, 9, 11, 12, 16, 17, 19], dtype='int64')
```

```
Prediction accuracy for the training dataset is:  84.1 %
Prediction accuracy for the testing dataset is:  81.7 %
Prediction balanced accuracy for the training dataset is:  75.5 %
Prediction balanced accuracy for the testing dataset is:  72.3 %
              precision    recall  f1-score   support

     class 1       0.8261    0.7125    0.7651        80
     class 2       0.8943    0.9091    0.9016       121
     class 3       0.6667    0.7273    0.6957        11
     class 4       0.9167    0.6111    0.7333        18
     class 5       0.4444    0.4444    0.4444         9
     class 6       0.9375    0.8824    0.9091        68
     class 7       0.8400    0.9545    0.8936        44
     class 8       0.8400    0.8400    0.8400        25
     class 9       0.6667    0.2667    0.3810        15
    class 10       0.5536    0.8857    0.6813        35
```

Συμπεραίνουμε οπότε πως έχοντας χειριστεί τα outliers η απόδοση του αλγορίθμου παρέμεινε σχετικά ίδια. Ωστόσο το f1 score για την κλάση 5 με λίγες παρατηρήσεις στο δείγμα έπαψε να ισούται με μηδέν όταν χρησιμοποιούσαμε τουλάχιστον 6 features ενώ στο dataset με τα outliers χρειαζόμασταν τουλάχιστον 9 features ώστε να 'ξεκινήσει' ο αλγόριθμος να αναγνωρίζει την συγκεκριμένη κλάση.

Μένει να δούμε αν το dataset όπου έχει προηγηθεί PCA(και noise reduction) δίνει περισσότερη απόδοση στον αλγόριθμο.

PCA Dataset(χωρίς outliers)

Θα ξεκινήσουμε να δούμε τι απόδοση μπορούμε να πετύχουμε με 2 features. Το feature '2' φαίνεται να έχει μεγάλη συσχέτιση με τη μεταβλητή στόχο (0.54) ωστόσο η μεταβλητότητα του κάθε feature φθίνει από την πρώτη προς την τελευταία.

```
In [27]: X.std()

Out[27]: 0     2.399302
         1     1.901893
         2     1.307347
         3     1.186794
         4     1.099384
         5     0.964001
         6     0.857259
         7     0.776926
         8     0.720554
         9     0.614085
         10    0.568327
```

Οπότε γενικά προτιμάμε μεν μεταβλητές συσχετισμένες με τη μεταβλητή στόχο ωστόσο προτιμάμε τις πρώτες. Χρησιμοποιώντας τη βιβλιοθήκη scikit learn καταλήγουμε:

```
selector=SelectKBest(mutual_info_classif,k=2)
selector.fit(X,Y)
selector.transform(X)
X.columns[selector.get_support()]
```

```
Index(['0', '2'], dtype='object')
```

Επιπλέον οι 0 μαζί με την 2 διατηρούν το 31%+9%=40% της αρχικής μεταβλητότητας των δεδομένων.

```
Prediction accuracy for the training dataset is:  49.4 %
Prediction accuracy for the testing dataset is:  50.5 %
Prediction balanced accuracy for the training dataset is:  34.1 %
Prediction balanced accuracy for the testing dataset is:  37.2 %
```

Η ακρίβεια του αλγορίθμου είναι 50.5% πολύ λιγότερη από αυτή που προκύπτει αντίστοιχα στην χωρίς PCA προσέγγιση με 2 features(67% !!).

Αναλύοντας διαφορετικούς συνδυασμούς features βγήκαν τα εξής αποτελέσματα:

#Features	Features	Pred.Acc.(Train)	Pred.Acc.(Test)	Bal.Acc (Train)	Bal.Acc (Test)
2	0,2	49.4	50.5	34.1	37.2
3	0,1,2	60.9	59.9	46.9	47.3
4	0,1,2,4	68.8	68.8	52.5	53.9
5	0,1,2,4,5	72.9	74.4	58.4	61.4
6*	0,1,2,3,4,5	77.7	78.1	67.8	68.5
7	0,1,2,3,4,5,7	82.2	78.9	69.7	66.6
8	0,1,2,3,4,5,8,9	84.3	81.7	74.8	71.2
9	0,1,2,3,4,5,7,8,10	85.5	83.1	77.1	72.9
10	0,1,2,3,4,5,7,8,9,10	88.4	83.8	79.6	74.6
11*	Πλήρες Μοντέλο#11	88,4	83,8	79.6	74,6

(*)Όλα τα f1-score είναι θετικά πλέον
(**)Η προσθήκη της 6 μας έδωσε 0% ακρίβεια .Άρα έχοντας τις υπόλοιπες μεταβλητές η 6 δεν δίνει καμία επιπλέον πληροφορία στον αλγόριθμο.

Αν και τα πρώτα δύο features είχανε μικρή απόδοση παρατηρούμε ότι η προσθήκη ενός feature ανεβάζει ραγδαία την απόδοση σε αντίθεση με το κανονικό dataset. Με 9 features ο αλγόριθμος έχει καταφέρει απόδοση 83.1 όπου από δω και πέρα φαίνεται ο αλγόριθμος να κάνει overfitting. Επιπρόσθετα ενώ στο κανονικό dataset χρειαζόμασταν τουλάχιστον 9 μεταβλητές ώστε όλες οι κλάσεις να έχουν θετικό recall το PCA dataset χρειάζεται 6.

Επιπλέον επαναλάβαμε την ανάλυση με PCA με 11 features σε dataset όπου δεν προηγήθηκε επεξεργασία των outliers και παρατηρήσαμε ότι η απόδοση του αλγορίθμου έπεσε ενώ παράλληλα εντοπίστηκε και overfitting issue.

```
Prediction accuracy for the training dataset is:  82.8 %
Prediction accuracy for the testing dataset is:  77.5 %
Prediction balanced accuracy for the training dataset is:  68.3 %
Prediction balanced accuracy for the testing dataset is:  66.3 %
--------------- Confusion Matrix ---------------
[[64  7  0  0  0  1  0  0  0  4]
 [10 91  0  0  0  2  0  0  0  0]
 [ 3  3  4  0  0  0  0  0  0  0]
 [ 0  3  0  9  0  0  0  0  0  0]
 [12  3  0  0  0  1  0  0  0  1]
 [ 1  3  0  0  0 53  8  0  0  0]
 [ 1  0  0  0  0  5 56  0  0  0]
 [ 0  0  0  0  0  1  2 19  0  0]
 [ 2  0  0  0  0  0  0  0  9  4]
 [17  1  0  0  0  0  0  0  1 25]]
```

Συμπεραίνουμε συνεπώς πως τα outliers επηρέασαν αρνητικά την ανάλυση στην κατηγοριοποίηση και επομένως θα προτιμήσουμε το dataset στο οποίο έχει προηγηθεί noise reduction. Επιπλέον καθώς η PCA επιταχύνει την διαδικασία του αλγορίθμου και οι αποδόσεις του ήταν όμοιες(ελαφρώς καλύτερα με PCA) προτείνεται το dataset όπου έχει προηγηθεί PCA και noise reduction.

Clustering

Θέλουμε να κάνουμε μη επιβλεπόμενη συσταδοποίηση στα δεδομένα. Αρχικά πρέπει να κάνουμε feature selection ωστόσο σε αντίθεση με την προηγουμένως κατηγοριοποίηση δε γνωρίζουμε τη μεταβλητή στόχο και επομένως δεν ξέρουμε ποιες μεταβλητές είναι πιο σημαντικές για τον στόχο. Στην επιβλεπόμενη συσταδοποίηση θα διαλέγαμε features με κριτήριο συσχετίσεις και μεγάλη διασπορά, συνεπώς ως μόνο κριτήριο για το feature selection απομένει η διασπορά. Οπότε αναζητούμε μεταβλητές με μεγάλη διασπορά αλλά παράλληλα να μην είναι αρκετά συσχετισμένες μεταξύ τους αφού αυτό θα σημαίνει πως δίνουν την 'ίδια' πληροφορία.

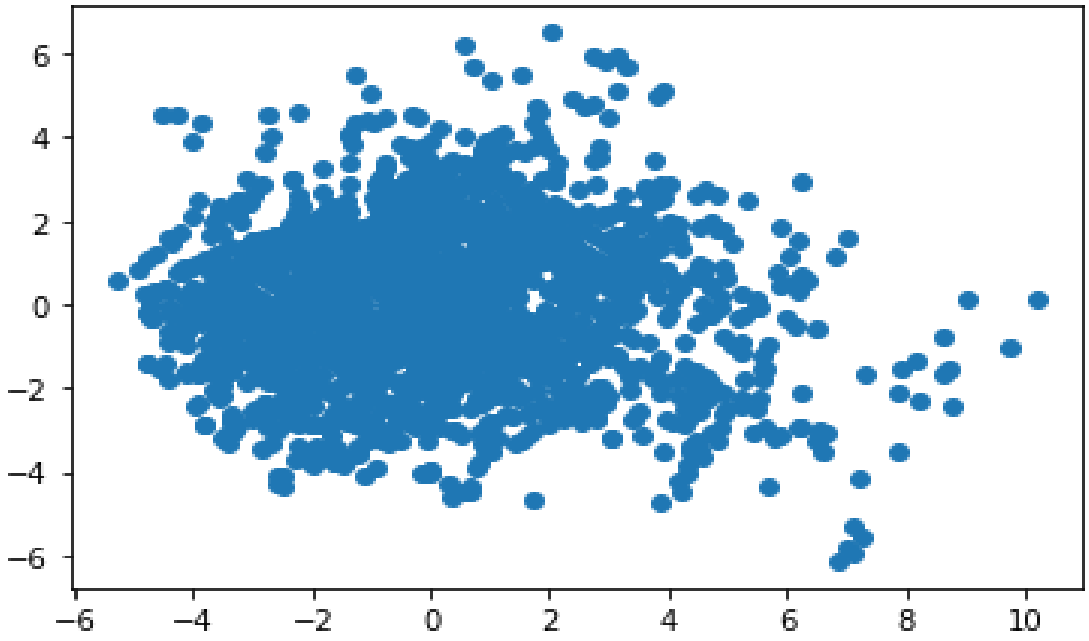
```
LB          9.840844
AC          0.003866
FM          0.046666
UC          0.002946
DL          0.002960
DS          0.000057
DP          0.000590
ASTV       17.192814
MSTV        0.883241
ALTV       18.396880
MLTV        5.628247
Width      38.955693
Min        29.560212
Max        17.944183
Nmax        2.949386
Nzeros      0.706059
Mode       16.381289
Mean       15.593596
Median     14.466589
Variance   28.977636
Tendency    0.610829
CLASS       3.026883
NSP         0.614377
```

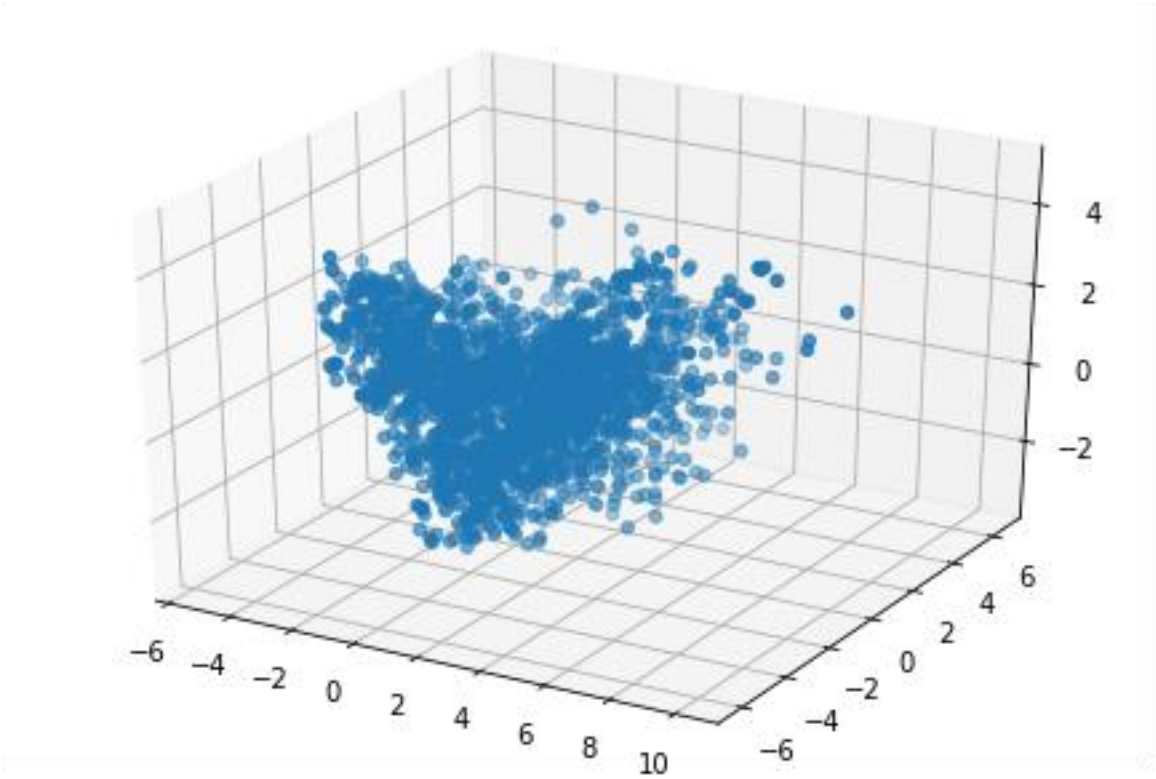
Πολλές από τις μεταβλητές να βλέπουμε να έχουν μεγάλη τυπική απόκλιση μπορούμε να δούμε στο προηγούμενο heatmap πως είναι αρκετά συσχετισμένες μεταξύ τους και ο συνδυασμός τριάδων προς εύρεση της καλύτερης για να σχηματίσουμε plots για να βρούμε patterv θα είναι μεγάλος σε αριθμό.\

Λαμβάνοντας υπόψιν όλα τα προαναφερθείσα θα κάνουμε PCA στα δεδομένα και θα πάρουμε τις 3 πρώτες συνιστώσες καθώς αυτές θα έχουν τη μεγαλύτερη διασπορά και θα διατηρήσουν το 45% της αρχικής μεταβλητότητας.
Στον πίνακα βλέπουμε το ποσοστό μεταβλητότητας των δεδομένων που έχει η κάθε συνιστώσα.

```
array([0.2885, 0.167 , 0.0869, 0.0713, 0.058 , 0.0486, 0.0468, 0.0441,
       0.0363, 0.0305, 0.0275, 0.0237, 0.0185, 0.0156, 0.0126, 0.0086,
       0.0063, 0.0056, 0.0024, 0.0013, 0.    ])
```

Θέλοντας τώρα να ‘δούμε’ τα δεδομένα θα δημιουργήσουμε ένα δισδιάστατο plot με τις πρώτες 2 συνιστώσες και ένα τρισδιάστατο με τις πρώτες 3.



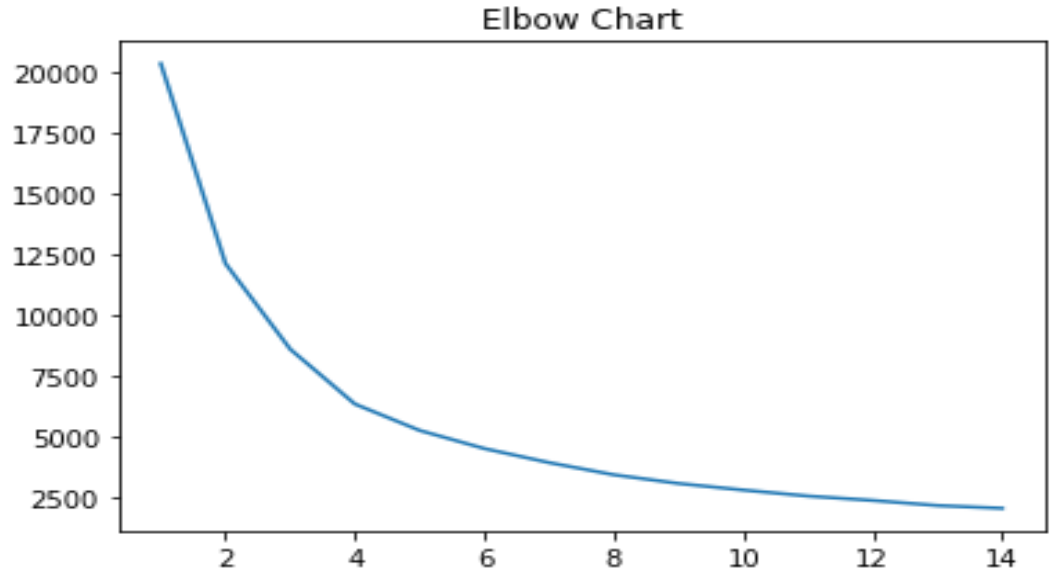


Στο πρώτο διάγραμμα θα περιμέναμε τα δεδομένα να είναι συμμετρικά γύρω από το 0 ως προς και τους 2 άξονες, καθώς με το PCA γίνεται και μετατόπιση στο 0, ωστόσο βλέπουμε ότι ως προς τον άξονα χ υπάρχει ένας συνωστισμός αρκετά δεξιά τα οποία μπορεί να αποτελούν outliers.Θα ερευνήσουμε αν έχουν σημαντικό ρόλο κατά την εξέλιξη της ανάλυσης και αν διαπιστωθεί ότι επηρεάζουν αρνητικά τη διαδικασία θα τα προσαρμόσουμε.

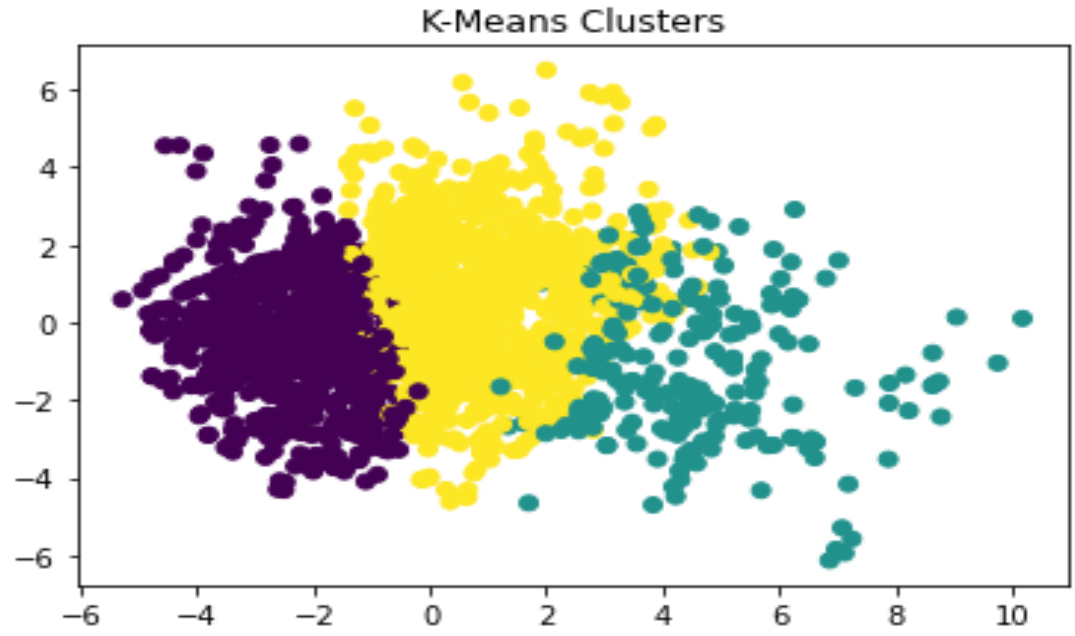
Επιπλέον και στα δύο διαγράμματα βλέπουμε πως το pattern είναι αρκετά πυκνό επομένως density based clustering algorithms δε θα είναι αποδοτικοί. Θα εφαρμόσουμε έναν distance based algorithm (K-Means) και έναν βασισμένο σε πιθανοφάνειες (Gaussian Mixture Models).

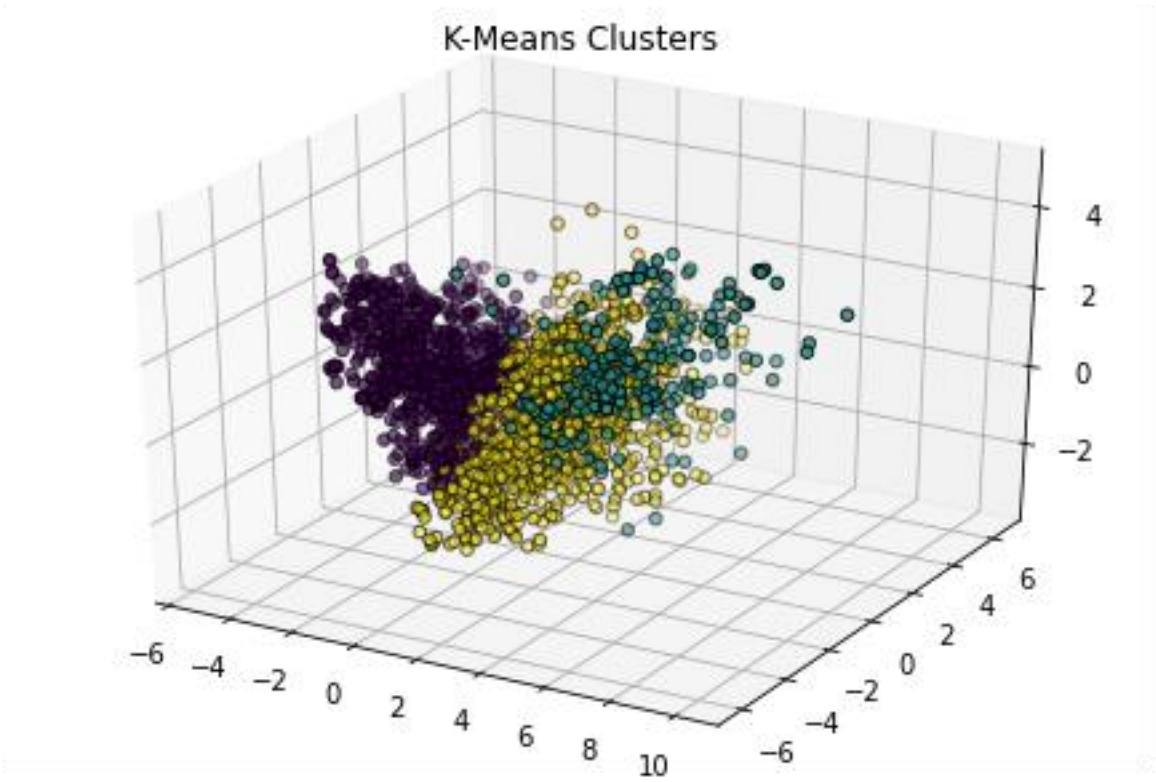
K-Means

Για την εφαρμογή του ο συγκεκριμένος αλγόριθμος πρέπει να γνωρίζει τον αριθμό των συστάδων που πρέπει να δημιουργήσει. Για την εύρεση του θα δούμε το Elbow Chart .

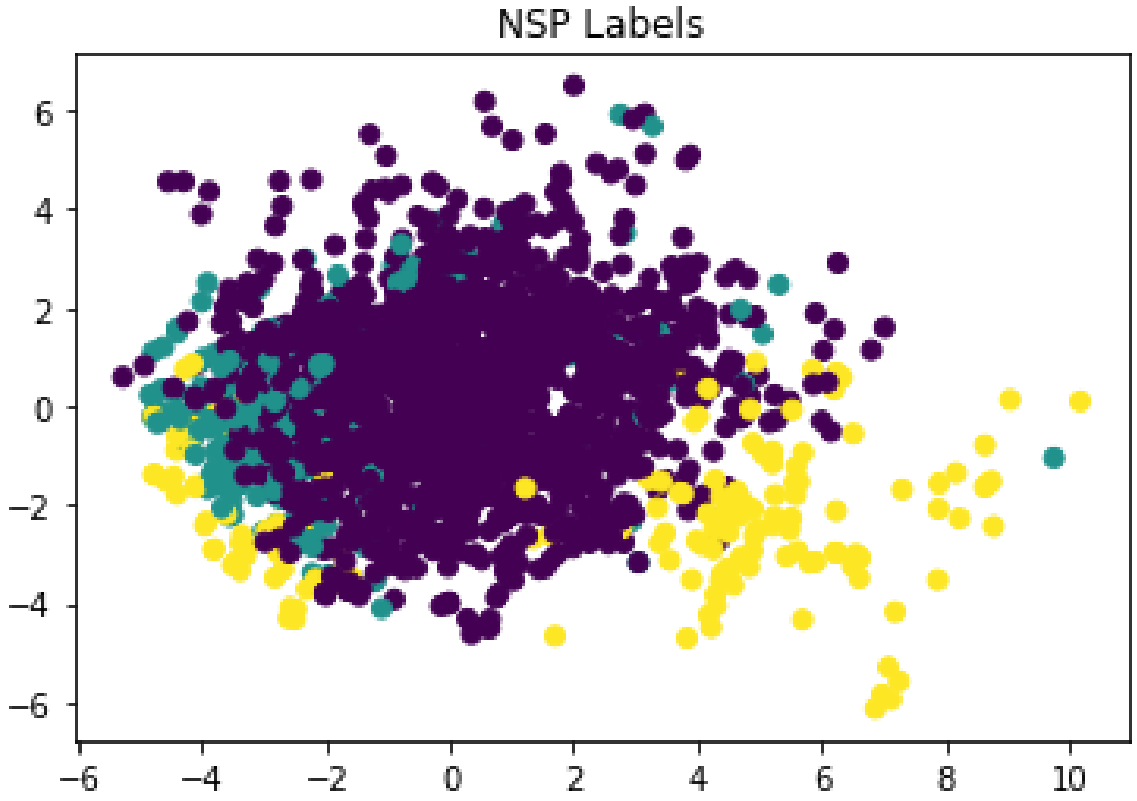


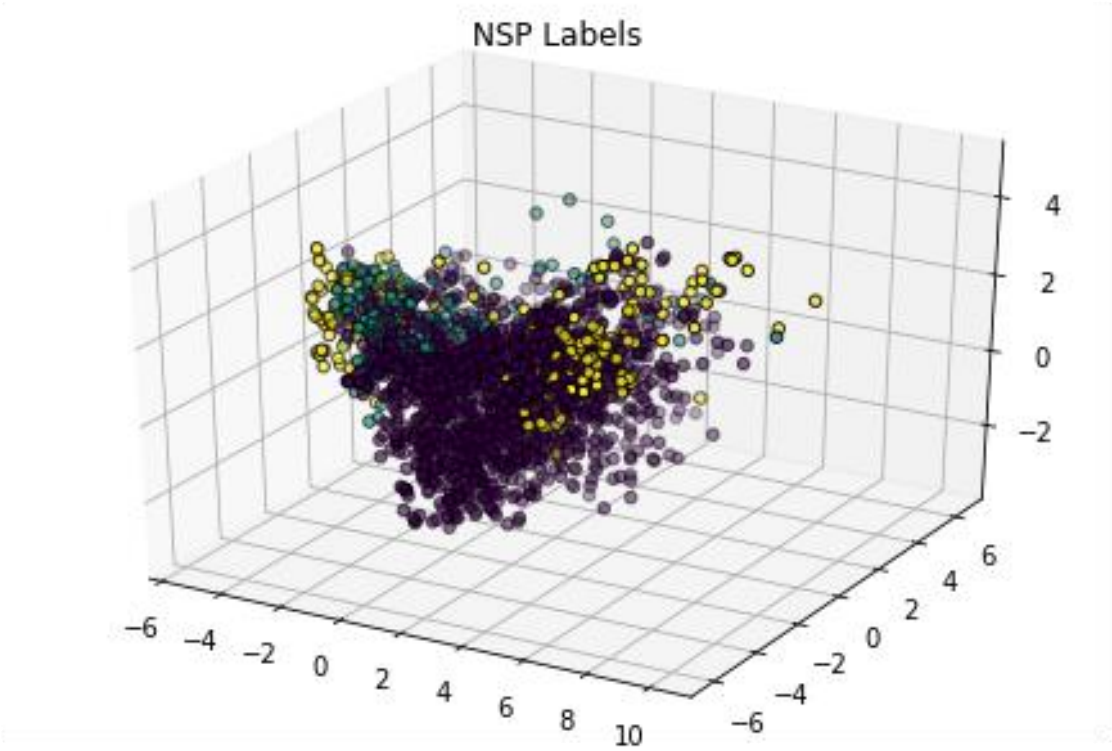
Στο διάγραμμα ο τελευταίος ‘αγκώνας’ βρίσκεται μεταξύ του 3 και του 4 ωστόσο θα προτιμήσουμε τη δημιουργία για σύγκριση στην συνέχεια της ανάλυσης.
Τρέχουμε οπότε τώρα τον αλγόριθμο και προκύπτουν τα εξής διαγράμματα.





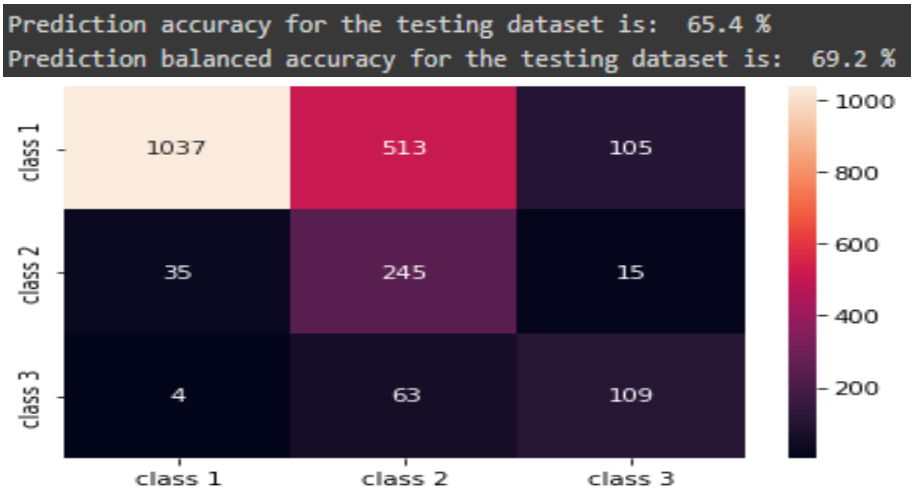
Ας τα συγκρίνουμε τώρα με τα αντίστοιχα διαγράμματα με labels την κατηγορία NSP.





Στο δισδιάστατο διάγραμμα του NSP βλέπουμε η μία κλάση βρίσκεται αριστερά, η άλλη εκτείνεται στο μεγαλύτερο μέρος του κέντρου ενώ η τρίτη απλώνεται κυρίως δεξιά με λίγα σημεία στο αριστερό μέρος. Δηλαδή ο άξονας χ που αποτελείται από την πρώτη συνιστώσα φαίνεται να έχει το μεγαλύτερο ρόλο ως προς τη μεταβλητή στόχο NSP. Επιπρόσθετα τα outliers στα δεξιά αποτελούν τη πλειοψηφία μιας κλάσης του NSP συνεπώς έχουνε σημαντική πληροφορία και ΔΕΝ πρέπει να απαλειφθούν/χειριστούν.

Θα δούμε τώρα πόσο όμοιες είναι οι 3 συστάδες που προέκυψαν μέσω του αλγορίθμου με τις 3 κλάσεις της NSP.



Οπότε χρησιμοποιώντας όλες τις 21 συνιστώσες(features) ο αλγόριθμος κατάφερε να κατανέμει σωστά το 65.4% των στοιχείων αν και από τα 1655 στοιχεία που ανήκουν στην κλάση 1 του NSP ο K-Means τους ομαδοποίησε στην αντίστοιχη της κλάσης 2 ενώ 105 στην αντίστοιχη της κλάσης 3.

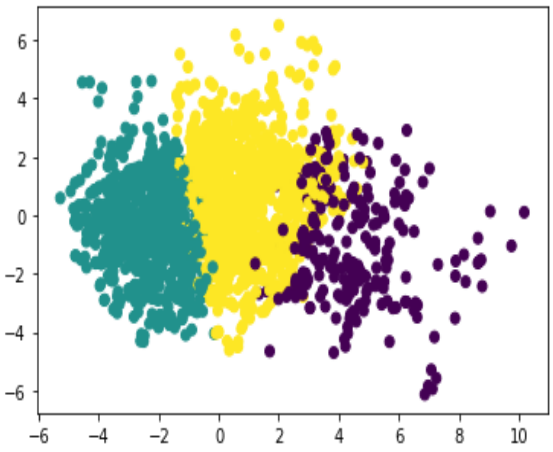
Θέλουμε τώρα να δούμε αν μπορούμε να καλυτερέψουμε την ακρίβεια του αλγορίθμου με διαφορετικό συνδυασμό features ή αν μπορούμε να επιτύχουμε κοντινή ακρίβεια με λιγότερα features.

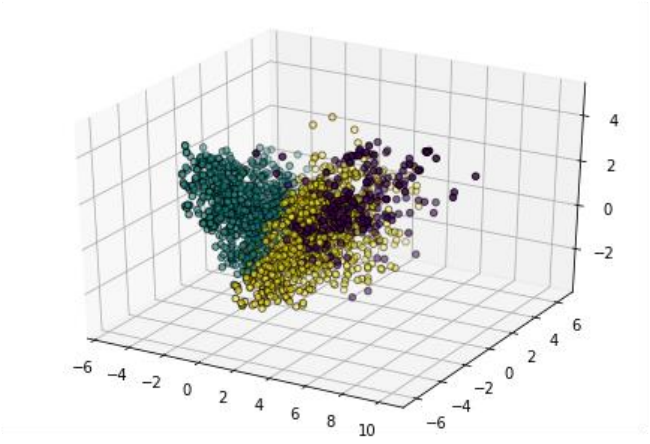
Καθώς τα features έχουνε φθίνουσα διασπορά μετά την ανάλυση τους σε Κύριες Συνιστώσες θα δοκιμάσουμε συνδυασμούς όπου κάθε φορά θα αφαιρούμε την τελευταία.
Οι 6 τελευταίες έχουνε στο σύνολο τους 2% μεταβλητότητα οπότε μπορούμε εύκολα να τις αφαιρέσουμε.

```
Prediction accuracy for the testing dataset is:  65.4 %
Prediction balanced accuracy for the testing dataset is:  69.2 %
--------------- Confusion Matrix ---------------
[[1037  513  105]
 [  35  245   15]
 [   4   63  109]]
```

Βλέπουμε πως οι 6 τελευταίες συνιστώσες δεν επηρέασαν καθόλου την απόδοση.

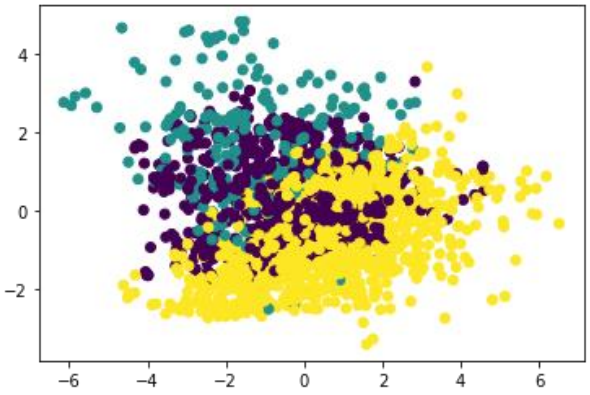
Κρατώντας μόνο τις 8 πρώτες συνιστώσες ο αλγόριθμος ‘διατήρησε’ την απόδοση του ενώ συνεχίζει να διαχωρίζει τις συστάδες ως προς τον άξονα χ (δηλαδή το πρώτο feature).

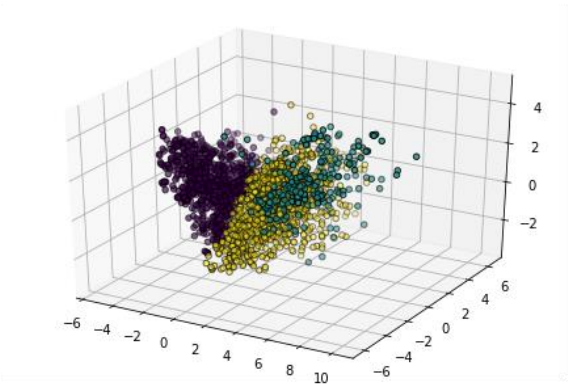





```
Prediction accuracy for the testing dataset is:  65.3 %
Prediction balanced accuracy for the testing dataset is:  69.2 %
--------------- Confusion Matrix ---------------
[[1034  514  107]
 [  35  245   15]
 [   4   63  109]]
```

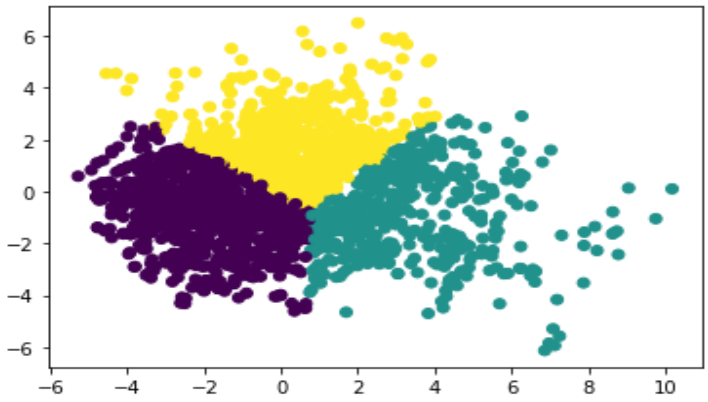
Συνεχίζοντας να αφαιρούμε features φθάνοντας να κρατάμε μόνο τις πρώτες 3 διατηρήθηκαν οι αποδόσεις αν και τα διαγράμματα αλλάξαν.





```
Prediction accuracy for the testing dataset is:  64.6 %
Prediction balanced accuracy for the testing dataset is:  69.2 %
--------------- Confusion Matrix ---------------
[[1016  520  119]
 [  31  249   15]
 [   1   66  109]]
```

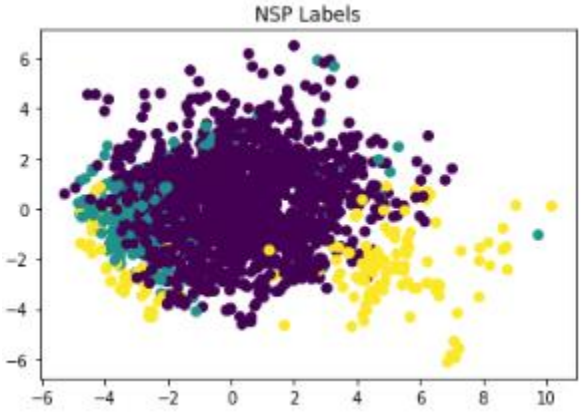
Πρώτη δραματική διαφορά εντοπίστηκε στο feature selection με μόνο τις πρώτες 2 μεταβλητές.

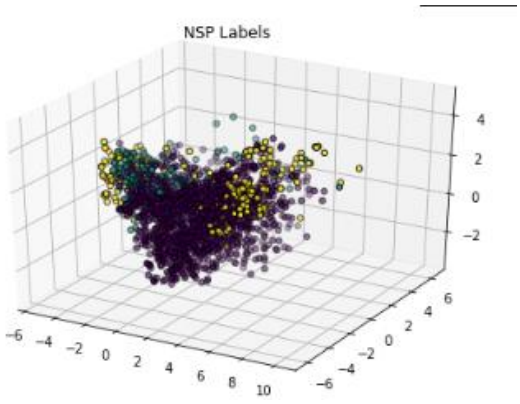


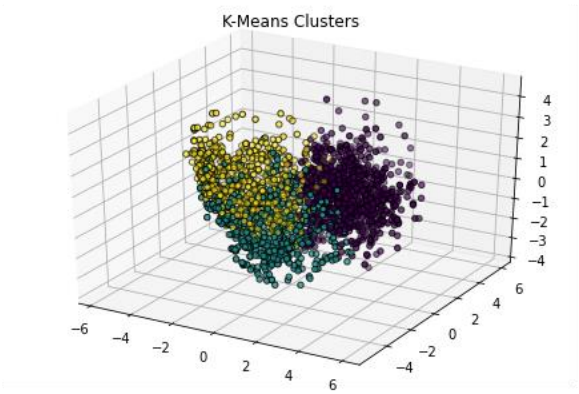
```
Prediction accuracy for the testing dataset is:  47.0 %
Prediction balanced accuracy for the testing dataset is:  60.8 %
--------------- Confusion Matrix ---------------
[[653 666 336]
 [ 46 237  12]
 [  0  66 110]]
```

Πέρα της πτώσης της ακρίβειας κατά 17% βλέπουμε επίσης σημαντική αλλαγή στο cluster plot καθώς μέχρι τώρα στην ανάλυση οι συστάδες κατανέμονταν αφηρημένα κυρίως ως προς τον άξονα y ενώ βλέπουμε το pattern να αλλάζει προς 3 ίσα κομμάτια ενός κύκλου, Επομένως η 3η συνιστώσα είναι σημαντική ως προς την μεταβλητή στόχος NSP.

Επιπλέον, αναφέραμε πριν την σημαντικότητα των outliers του συγκεκριμένου δείγματος στην συσταδοποίηση οπότε επαναλήφθηκε η προηγούμενη ανάλυση στο δείγμα αφού πρώτα έγινε noise reduction.







Επιπλέον διατηρώντας και τα 21 features η απόδοση του αλγορίθμου ήταν αρκετά χαμηλότερη. Οι ισχυρισμοί μας για τα outliers επιβεβαιώθηκαν.

```
Prediction accuracy for the testing dataset is:  46.2 %
Prediction balanced accuracy for the testing dataset is:  49.4 %
--------------- Confusion Matrix ---------------
[[696 461 498]
 [ 29 246  20]
 [ 90  46  40]]
```

Τέλος κάναμε την ίδια ανάλυση και στα αρχικά δεδομένα όπου δεν προηγήθηκε PCA αλλά απλώς scaling και διατηρώντας όλα τα 21 features προκύψαν τα εξής αποτελέσματα.

```
Prediction accuracy for the testing dataset is:  65.4 %
Prediction balanced accuracy for the testing dataset is:  69.2 %
--------------- Confusion Matrix ---------------
[[1037  513  105]
 [  35  245   15]
 [   4   63  109]]
```

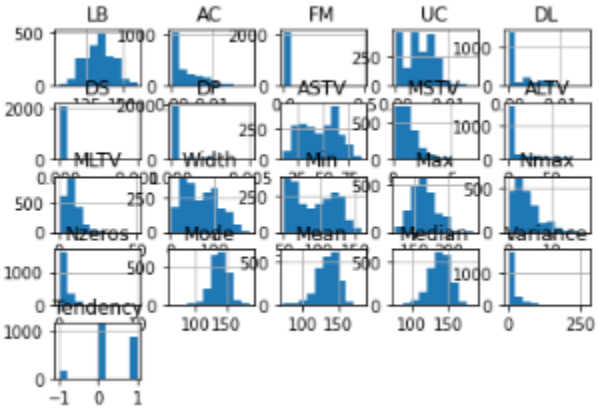
Διαπιστώνουμε οπότε ότι οι αποδόσεις όταν χρησιμοποιούμε και στα 2 datasets όλα τα features δεν έχουν μεγάλη διαφορά το πλεονέκτημα με το dataset που έχει προηγηθεί PCA σε μη επιβλεπόμενη ομαδοποίηση είναι πως μπορούμε ευκολότερα να κάνουμε feature selection κρατώντας τις ‘πλούσιες’ σε πληροφορία πρώτες συνιστώσες ενώ στο κανονικό dataset θα έπρεπε ‘τυφλά’ να δοκιμάζουμε μεγάλο στο πλήθος συνδυασμούς και δύσκολο να καταφέρναμε την ίδια απόδοση διατηρώντας μόνο 3 από τα 21 features.
Έτσι ο αλγόριθμος καταφέρνει ίδια αποτελέσματα μεν αλλά με περισσότερη ταχύτητα.

Gaussian Mixture Models

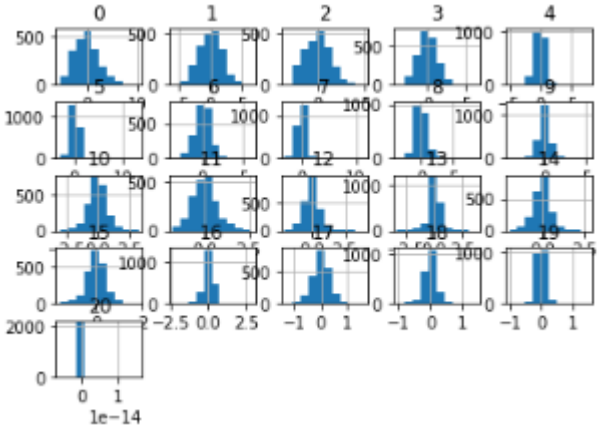
Σε αντίθεση με τον K-Means ο αλγόριθμος GMM δεν δημιουργεί clusters με κριτήριο την απόσταση αλλά τις κατανομές των στοιχείων. Ένα πλεονέκτημα είναι πως τα δεδομένα δε χρειάζεται να παίρνουν κάποιο shape για να είναι αποδοτικός και έχοντας δει το plot των δεδομένων ο GMM αποτελεί καλή επιλογή.

Επιπλέον, αν και αποτελεί κυρίως προτίμηση παρά αναγκαιότητα, ο GMM προτιμά dataset που προέρχονται από την κανονική κατανομή. Επιπλέον καθώς ο αλγόριθμος βασίζεται σε διασπορά και έχουμε μη επιβλεπόμενη ομαδοποίηση θα εργαστούμε με το dataset που έχει προηγηθεί PCA.

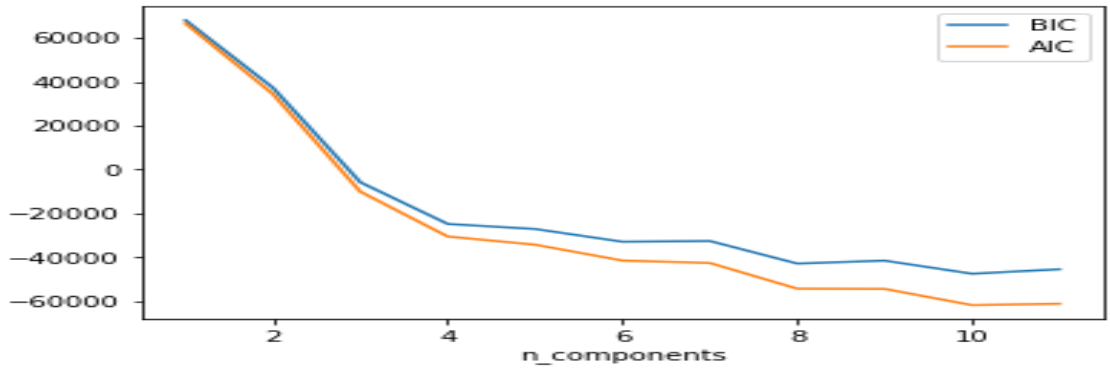
Before PCA



After PCA

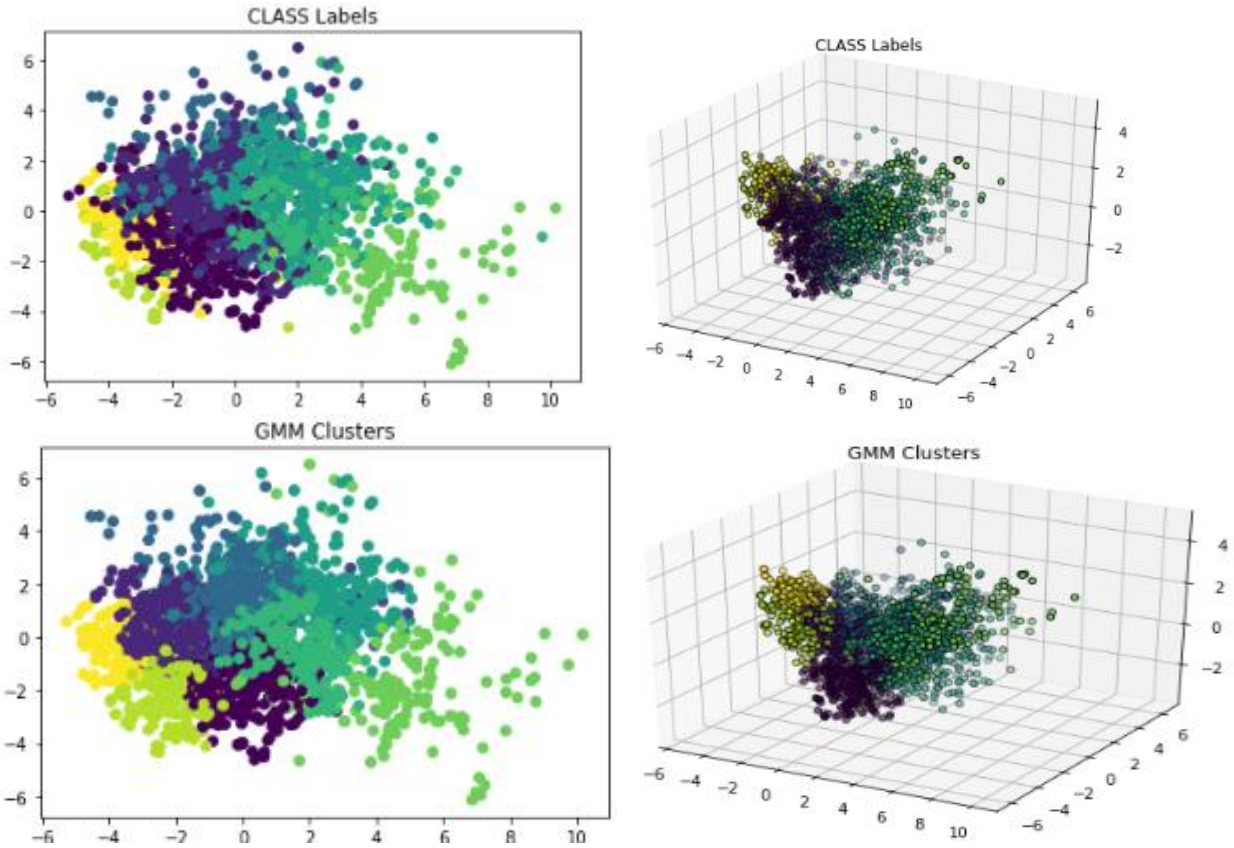


Επόμενο βήμα είναι να καθοριστεί ο αριθμός κ των συστάδων που θέλουμε. Θα δούμε τα διαγράμματα με τις τιμές των AIC/BIC για κ από 1 μέχρι 11.



Βλέπουμε πως η καμπύλη κατεβαίνει απότομα μέχρι το 3 και στην συνέχεια φθίνει σταδιακά μέχρι το 10 που παίρνει την χαμηλότερη τιμή. Θα εφαρμόσουμε λοιπόν τον GMM για κ=10.

Ας δούμε τώρα το διάγραμμα του GMM με τις 10 συστάδες και το αντίστοιχο για το dataset με τη μεταβλητή στόχο 'CLASS' η οποία έχει 10 κλάσεις χρησιμοποιώντας όλα τα features αρχικά..



Γενικά είναι αρκετά δύσκολο να δούμε την ακρίβεια μέσω διαγραμμάτων σε ένα αρκετά πυκνό plot και με τόσες κλάσεις. Ας δούμε επομένως το confusion matrix.

Παρατηρούμε αρκετές λάθος προβλέψεις όπου οδηγούν σε κακή ακρίβεια του αλγορίθμου.

```
Prediction accuracy for the testing dataset is:  20.5 %
Prediction balanced accuracy for the testing dataset is:  44.7 %
```

Δοκιμάζοντας για διαφορετικές παραμέτρους τον αλγόριθμο η ακρίβεια δεν βελτιώθηκε.

Θα δούμε οπότε τι συμβαίνει για κ=3 με όλα τα features και με τα 4 διαφορετικά covariance types του GMM.

Full Covariance

```
Prediction accuracy for the testing dataset is:  55.8 %
Prediction balanced accuracy for the testing dataset is:  63.0 %
--------------- Confusion Matrix ---------------
[[850 708  97]
 [ 45 235  15]
 [  8  66 102]]
```

Tied Covariance

```
Prediction accuracy for the testing dataset is:  47.5 %
Prediction balanced accuracy for the testing dataset is:  42.4 %
--------------- Confusion Matrix ---------------
[[782 859  14]
 [ 76 216   3]
 [105  59  12]]
```

Diag Covariance

```
Prediction accuracy for the testing dataset is:  56.1 %
Prediction balanced accuracy for the testing dataset is:  60.0 %
--------------- Confusion Matrix ---------------
[[893 622 140]
 [ 81 194  20]
 [ 33  37 106]]
```

Spherical Covariance

```
Prediction accuracy for the testing dataset is:  66.9 %
Prediction balanced accuracy for the testing dataset is:  69.1 %
--------------- Confusion Matrix ---------------
[[1078  428  149]
 [  34  235   26]
 [   4   62  110]]
```

Είναι φανερό πως ο GMM προτιμάει την spherical covariance στο dataset μας με 66.9%. Η απόδοση του GMM είναι μακράν καλύτερη στην κατηγορία NSP ωστόσο παραμένει χαμηλή και χρονοβόρα καθώς έχει όλα τα features.

Επομένως θα κάνουμε feature selection με σκοπό να αυξήσουμε την απόδοση ή τουλάχιστον να κάνουμε το μοντέλο πιο οικονομικό.

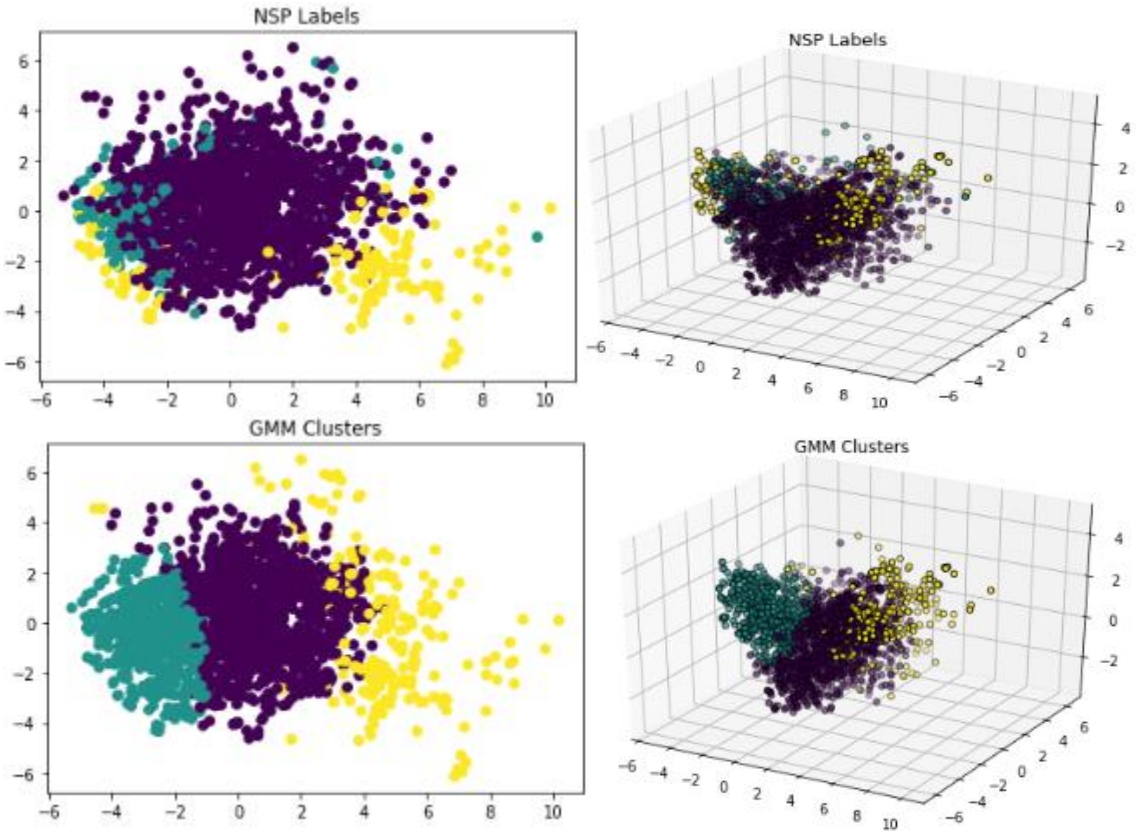
Με τα πρώτα δύο features που έχουνε και τη μεγαλύτερη διασπορά προκύψαν τα εξής:

```
Prediction accuracy for the testing dataset is:  47.6 %
Prediction balanced accuracy for the testing dataset is:  41.7 %
--------------- Confusion Matrix ---------------
[[871 555 229]
 [255  31   9]
 [ 67   0 109]]
```

Παρατηρούμε πτώση της τάξης 20% συνεπώς θα προσθέσουμε επιπλέον features. Την επόμενη μεγαλύτερη διασπορά έχει η 3η συνιστώσα.

```
Prediction accuracy for the testing dataset is:  81.5 %
Prediction balanced accuracy for the testing dataset is:  72.2 %
--------------- Confusion Matrix ---------------
[[1416  209   30]
 [  78  212    5]
 [  12   60  104]]
```

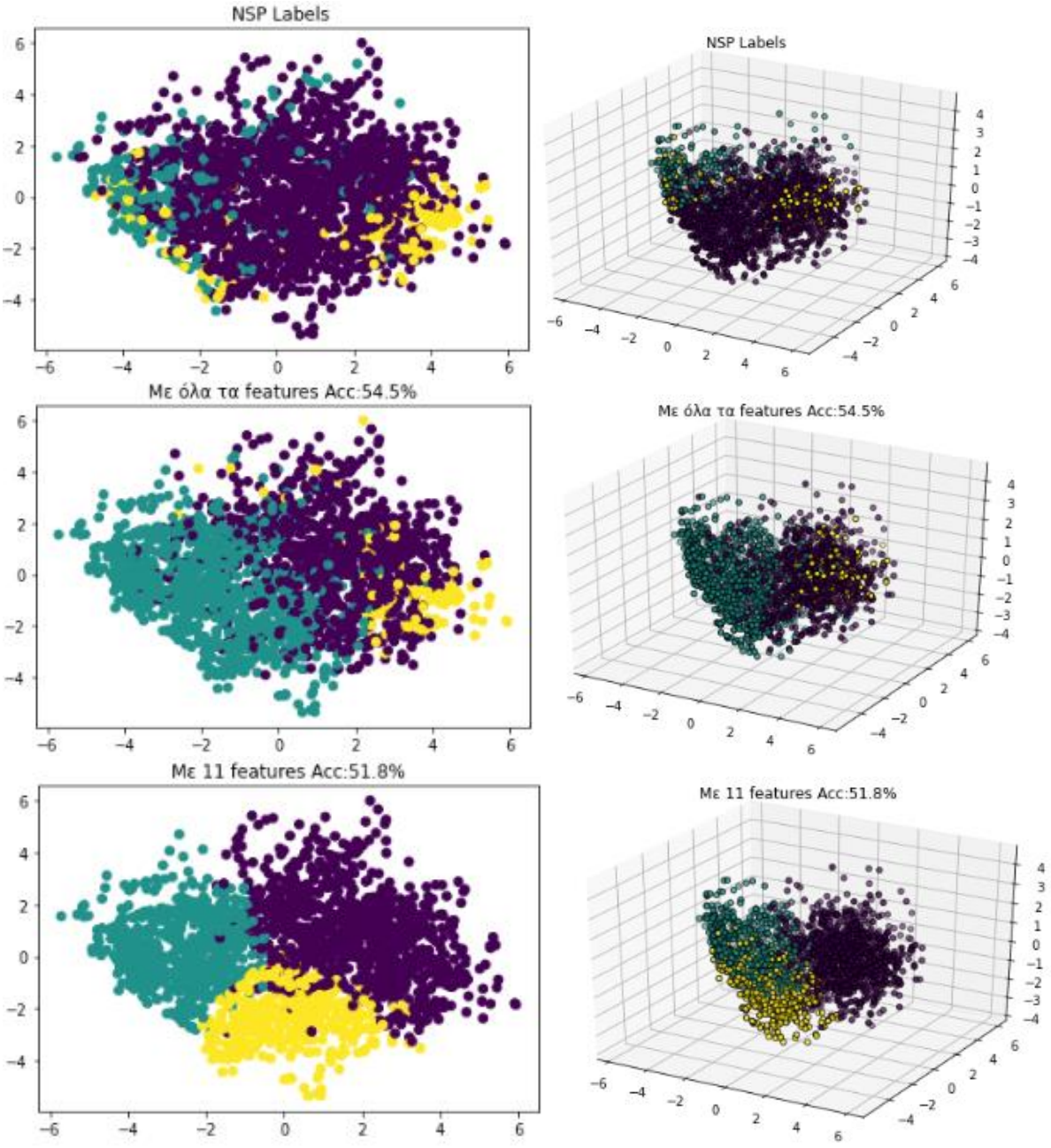
Με την προσθήκη της 3ης συνιστώσας βλέπουμε τεράστια άνοδο στην ακρίβεια που φτάνει το 81.5%.

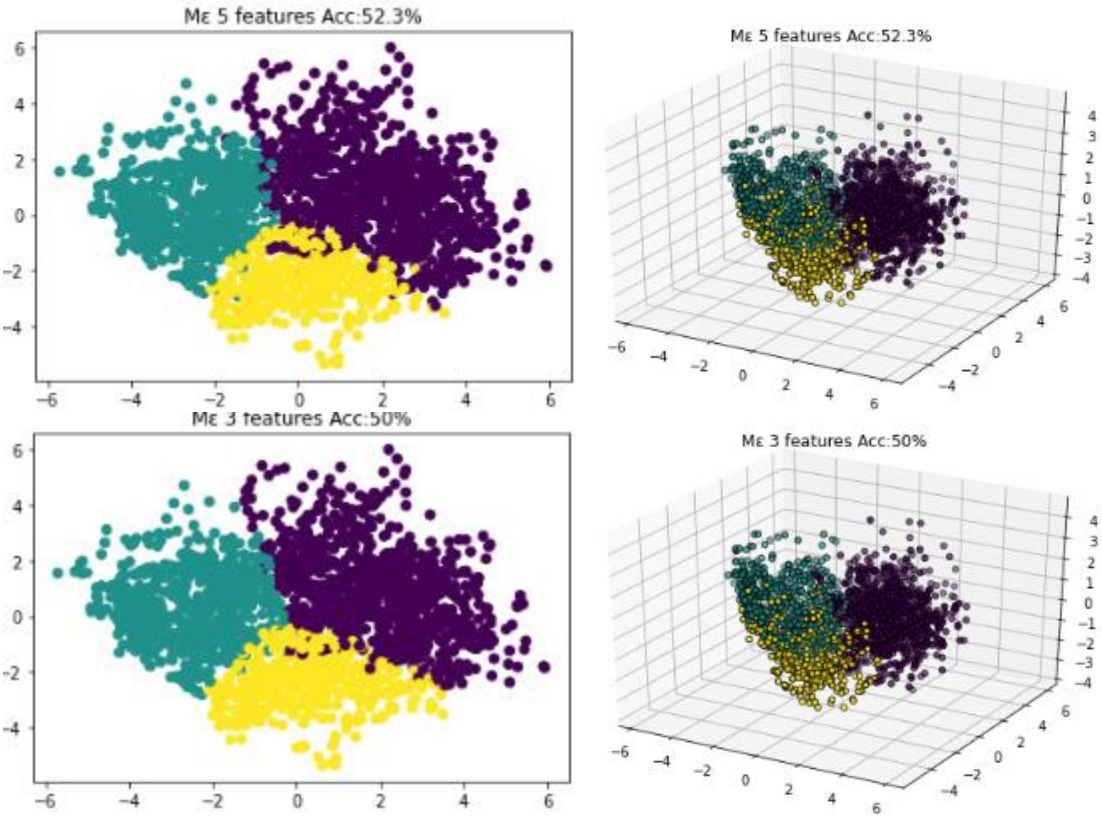


Βλέπουμε η συστάδα δεξιά πιάνει τα περισσότερα στοιχεία της αντίστοιχης κλάσης της NSP και μερικά από την κεντρική ενώ η αριστερή παίρνει επιπλέον κάποια στοιχεία από την κεντρική (κλάση 1 της NSP). Αυτό φαίνεται και στο confusion matrix καθώς 209 στοιχεία της κλάσης 1 της NSP μπήκαν στην 2η συστάδα του GMM και 30 στην 3η συστάδα.

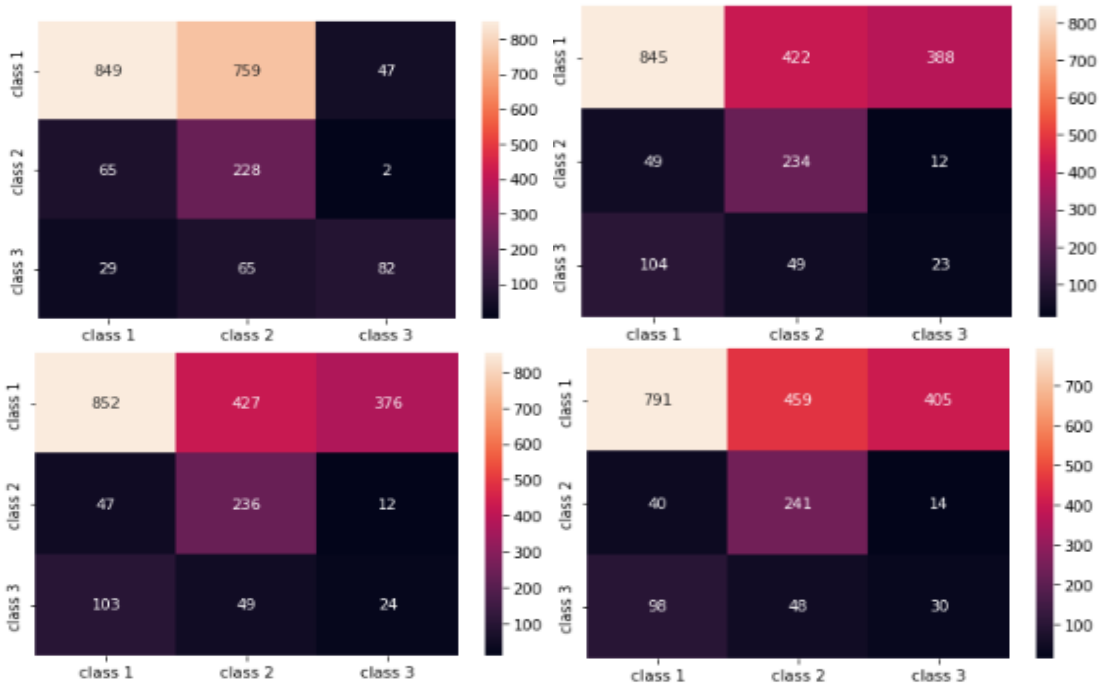
Συνεχίζοντας την ανάλυση προσθέτοντας παραπάνω features στο προηγούμενο μοντέλο βλέπουμε πως η ακρίβεια φθίνει επομένως καταλήγουμε ότι το βέλτιστο μοντέλο είναι αυτό με τα πρώτα 3 features και ακρίβεια 81.5%.

Επαναλάβαμε την ανάλυση στο dataset όπου έχει προηγηθεί noise reduction(και PCA) και όπως στη συσταδοποίηση με τον K-Means η απόδοση του αλγορίθμου έπεσε σημαντικά αφού πολλά από τα outliers χαρακτηρίζουν κλάση του NSP.





Από τα plots διαπιστώνουμε πως το plot του πλήρους μοντέλου των συστάδων διαφέρει πολύ από το αντίστοιχο της NSP και τα διαγράμματα με διαφορετικό συνδιασμό features δε καταφέρνουν κάτι καλύτερο. Επιπλέον βλέπουμε πως καθώς τα features μειώνονται ο αλγόριθμος τείνει να δημιουργεί ίσα κομμάτια μιας σφαίρας όμοια με του K-Means.



```
np.unique(Y, return_counts=True)
```

```
(array([1, 2, 3]), array([1655,  295,  176]))
```

Ωστόσο επειδή βλέποντας στον πάνω πίνακα το πλήθος των στοιχείων που ανήκουν στην κάθε κλάση του NSP το γεγονός ότι καθώς ενώ μειώνουμε τα features του αλγορίθμου ο GMM σταδιακά τείνει να συσταδοποιεί σε 3 ‘ίσες’ συστάδες οδηγεί στο πρόβλημα που βλέπουμε στα confusion matrix όπου πολλά στοιχεία της κλάσης 1 της NSP δεν ‘χωράνε’ στη αντίστοιχη συστάδα του GMM.

Στην ανάλυση του K-Means είχαμε δει ότι σημεία που είχαν μεγάλη απόσταση δεξιά από τον συνωστισμό αποδείχθηκαν κρίσιμα για την ορθή λειτουργία του αλγορίθμου. Επομένως καθώς απόσταση ερμηνεύεται ως και μεγάλη διασπορά και ο GMM βασίζεται στην διασπορά του δείγματος ήταν αναμενόμενο πως η επεξεργασία των outliers στο συγκεκριμένο dataset είχε ζημιώδη αποτελέσματα.

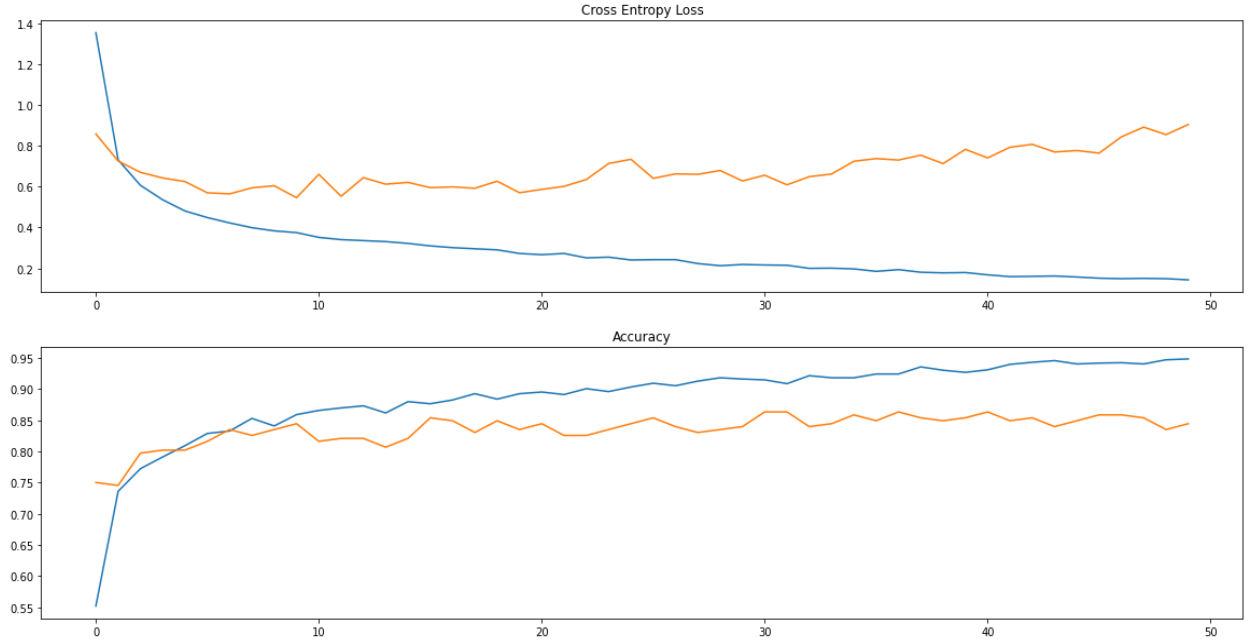
Συμπεραίνουμε συνεπώς πως τα outliers του dataset έχουν χρήσιμη πληροφορία και δε πρέπει να απαληφθούν.

Classification με Νευρωνικά Δίκτυα

Θέλουμε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο MLP το οποίο θα μαθαίνει από τα 21 χαρακτηριστικά και θα πραγματοποιεί κατηγοριοποίηση στα 10 μορφολογικά μοτίβα (FHR).

Είδαμε στο ιστόγραμμα στην αρχή της ανάλυσης πως τα δεδομένα μας δείχνουνε να είναι κανονικά επομένως θα χρησιμοποιήσουμε τυποποίηση για να κάνουμε scaling το dataset.

Χρησιμοποιώντας 50 νευρώνες για την πρώτη κρυφή στοιβάδα και 10 για τη δεύτερη με όλα τα features προέκυψαν τα εξής:

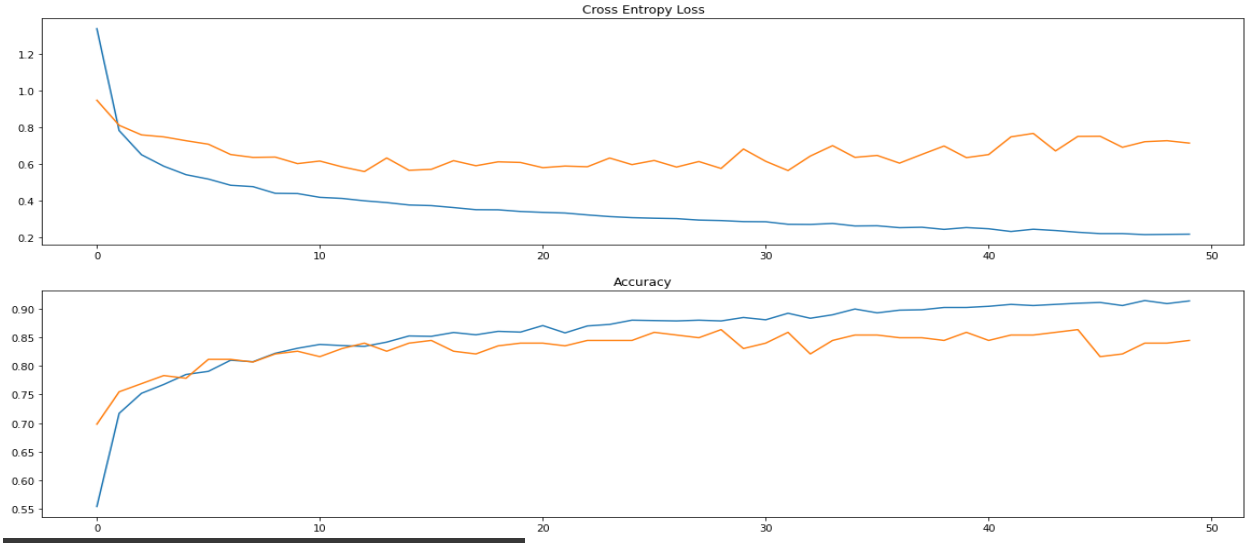


Βλέπουμε πως η κίτρινη καμπύλη που αντιστοιχεί στο Cross Entropy Loss του validation data να μένει σταθερή κοντά στο 0.6 και να απέχει αρκετά από την καμπύλη του training data που δείχνει να φθίνει προς το 0.

```
Train accuracy: 94.76%
Train Accuracy by model.predict: 94.76%
Val Accuracy by model.predict: 84.43%
Test Accuracy by model.predict: 82.39%
Train MMC:  0.9384900482119057
Val MMC:  0.8138546120947644
Test MMC:  0.7907411411222297
```

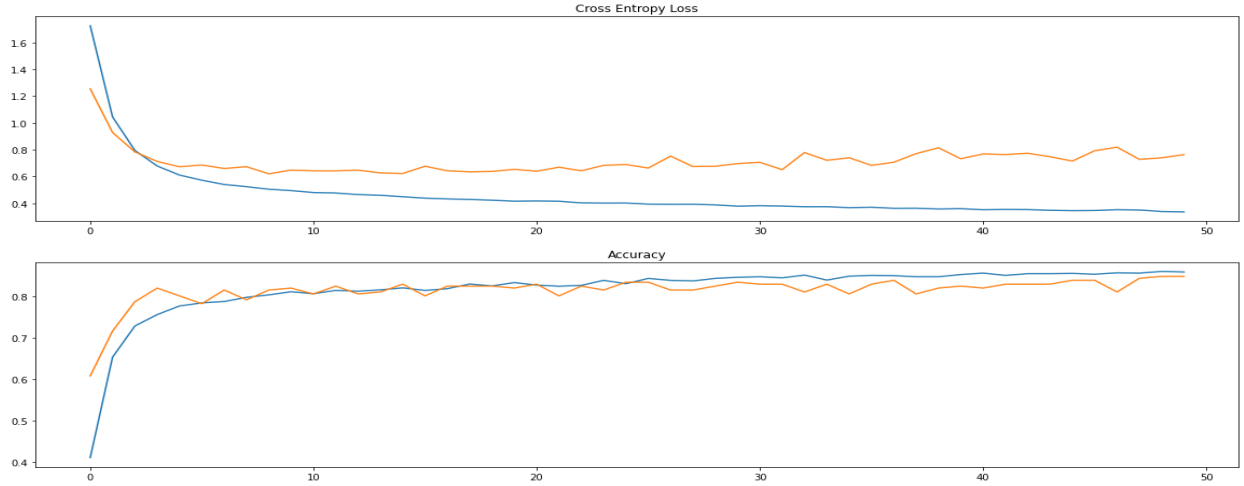
Η ακρίβεια του αλγόριθμου στο test dataset είναι 82.39%, και με Matthews correlation coefficient ίσο 0.79 αριθμοί αρκετά ικανοποιητικοί οπότε μένει να δούμε αποδόσεις για διαφορετικούς αριθμούς νευρώνων.

Με χρήση 10/50 νευρώνων:




```
Train accuracy: 92.07%
Train Accuracy by model.predict: 92.07%
Val Accuracy by model.predict: 84.43%
Test Accuracy by model.predict: 80.28%
Train MMC:  0.9063008014958808
Val MMC:  0.8104932857283526
Test MMC:  0.767609533768811
```

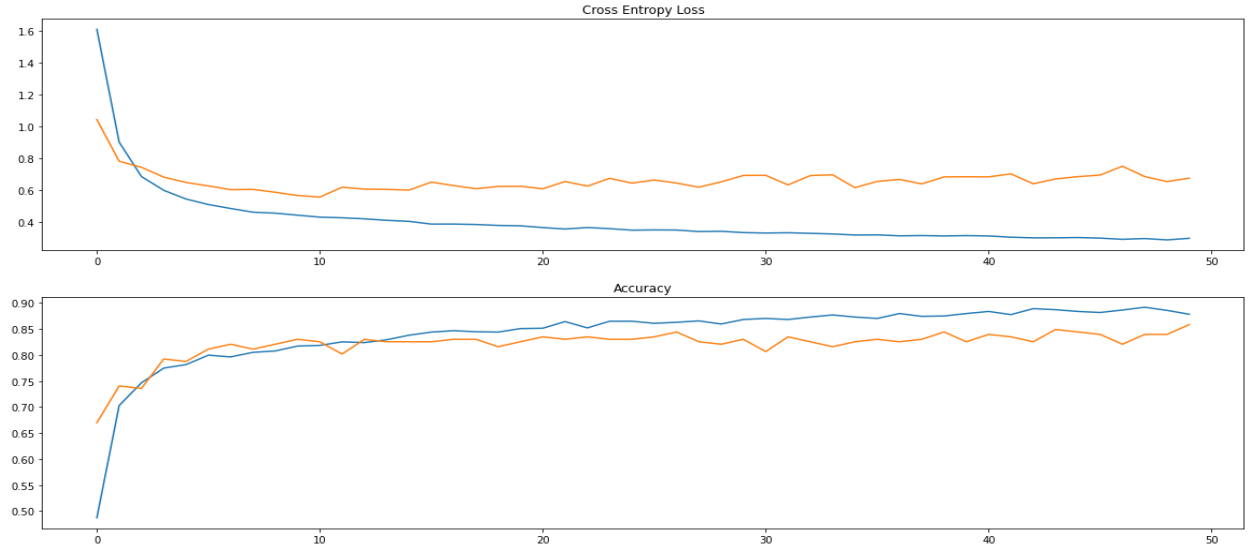
Με χρήση 10/10 νευρώνων:




```
Train accuracy: 86.76%
Train Accuracy by model.predict: 86.76%
Val Accuracy by model.predict: 84.91%
Test Accuracy by model.predict: 82.16%
Train MMC:  0.8425149355574418
Val MMC:  0.816982172747593
Test MMC:  0.7872289266963235
```

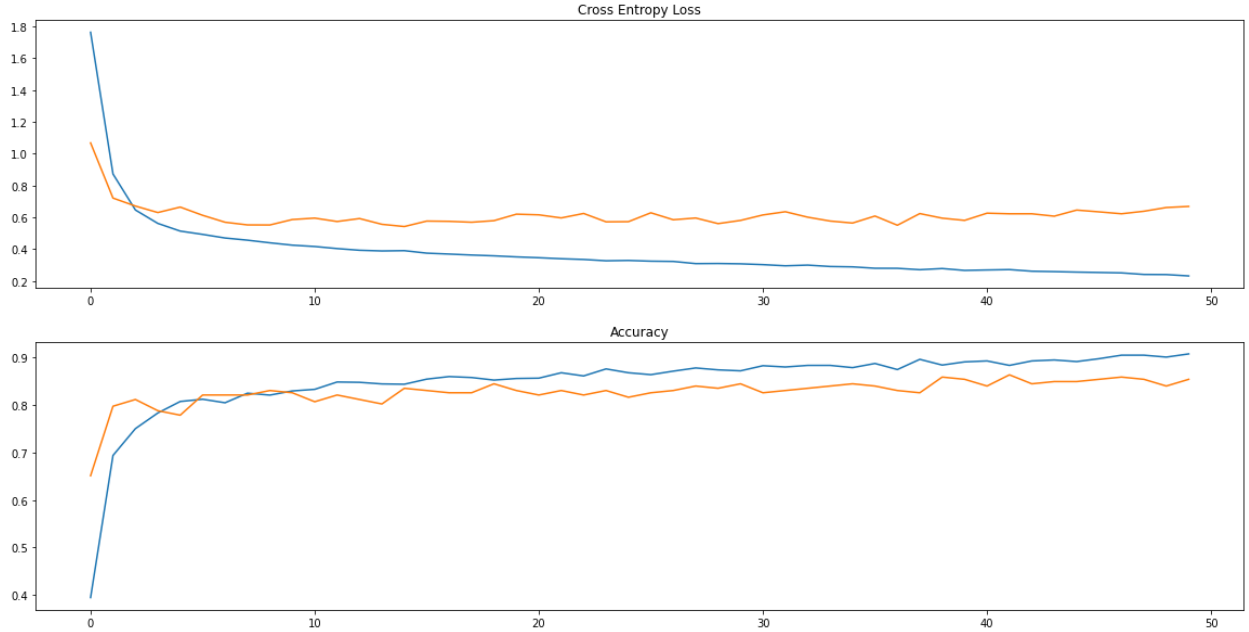
Παρατηρούμε ότι προσθέτοντας 40 νευρώνες στην πρώτη στοιβάδα (δηλαδή το 50/10) δεν κερδίζουμε σημαντική ακρίβεια στο test data παρά μόνο στο training data.

Με χρήση 20/10 νευρώνων:



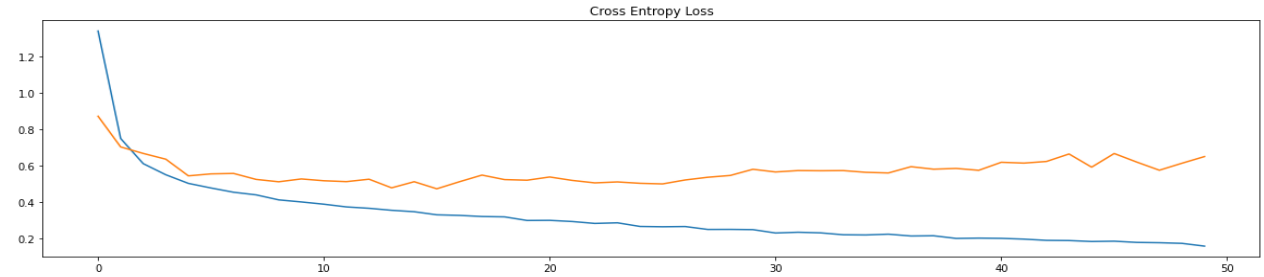

```
Train accuracy: 89.31%
Train Accuracy by model.predict: 89.31%
Val Accuracy by model.predict: 85.85%
Test Accuracy by model.predict: 83.33%
Train MMC:  0.8736938842575073
Val MMC:  0.8278375369778571
Test MMC:  0.8022617815141841
```

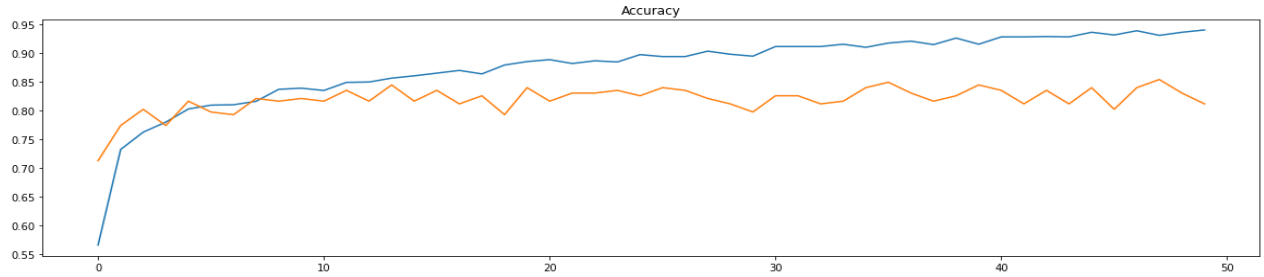
Με χρήση 10/20 νευρώνων:




```
Train accuracy: 89.92%
Train Accuracy by model.predict: 89.92%
Val Accuracy by model.predict: 85.38%
Test Accuracy by model.predict: 81.46%
Train MMC:  0.8812283893335561
Val MMC:  0.8248836774704852
Test MMC:  0.7803318828557291
```

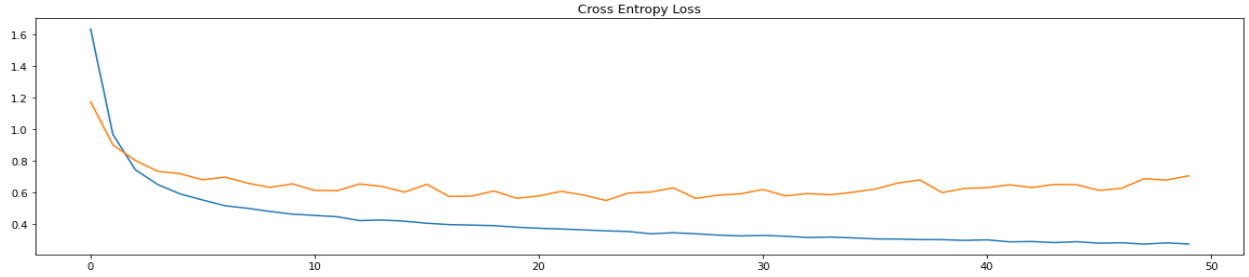
Με χρήση 20/20 νευρώνων:

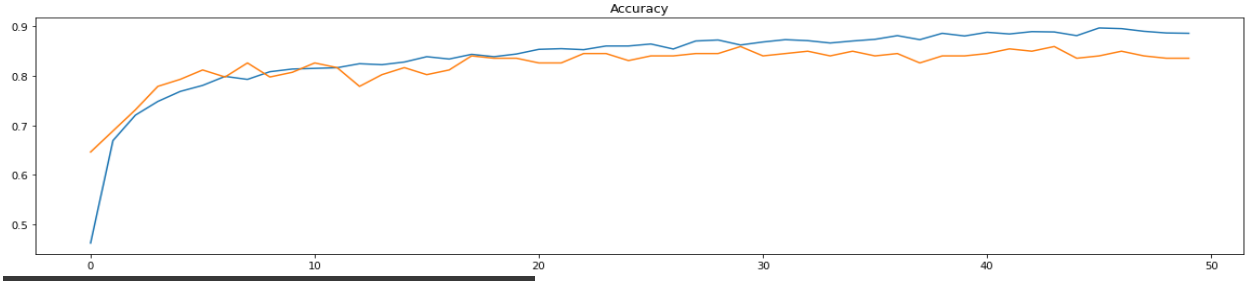





```
Train accuracy: 93.95%
Train Accuracy by model.predict: 93.95%
Val Accuracy by model.predict: 81.13%
Test Accuracy by model.predict: 81.92%
Train MMC:  0.9284082378230571
Val MMC:  0.7716061133209865
Test MMC:  0.7848777748863548
```

Με χρήση 15/15 νευρώνων:

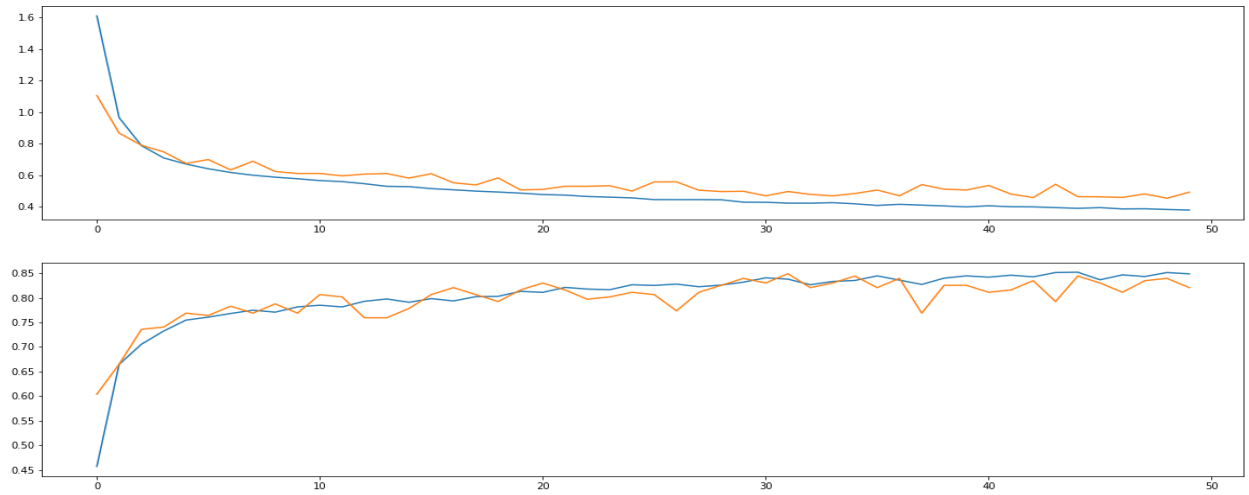





```
Train accuracy: 90.73%
Train Accuracy by model.predict: 90.73%
Val Accuracy by model.predict: 83.49%
Test Accuracy by model.predict: 82.16%
Train MMC:  0.8902157881079056
Val MMC:  0.8017834114899375
Test MMC:  0.7883872583637899
```

Διαπιστώνουμε πως μετά τους 20 νευρώνες κυρίως η ακρίβεια στο training data ανεβαίνει αισθητά ενώ η αντίστοιχα το testing data δεν εντοπίζεται σημαντική διαφορά.. Επιπρόσθετα, προσθήκη επιπλέον νευρώνων στην πρώτη κρυφή στοιβάδα δείχνουν να αποδίδουν περισσότερο από προσθήκη στην δεύτερη (έχοντας τουλάχιστον 10). Επομένως θα επιλέξουμε 20 νευρώνες στην πρώτη και 10 στην δεύτερη αφού με περεταίρω νευρώνες ο αλγόριθμος δείχνει να μαθαίνει υπερβολικά καλά τον θόρυβο στο training dataset χωρίς να συνεισφέρει στο test dataset.

Θα εφαρμόσουμε τον αλγόριθμο με τις ίδιες παραμέτρους στο dataset όπου έχει προηγηθεί preprocessing (εξομάλυνση /όχι αφαίρεση) για τα outliers.



```
Train accuracy: 86.42%
Val Accuracy by model.predict: 82.08%
Test Accuracy by model.predict: 82.63%
Train MMC:  0.838767948759335
Val MMC:  0.7870349046309121
Test MMC:  0.7938592420469803
```

Συμπεραίνουμε πως ο επιπλέον θόρυβος εξαιτίας των outliers που υπήρχε στην αρχική ανάλυση δεν επηρέασε σημαντικά την απόδοση του αλγορίθμου στο testing data (83.33% έναντι 82.63%) και απλά ανέβασε το train accuracy του αλγορίθμου (89.92% έναντι 86.42%) ενώ το Matthews correlation score και των δύο είναι κοντά στο 0.80. Ωστόσο παρατηρούμε πως το cross entropy του validation data είναι χαμηλότερο και πιο κοντά στου training data. Δηλαδή ο θόρυβος οδήγησε σε overfitting μικρού βαθμού.

Θα δοκιμάσουμε τώρα την ανάλυση με λιγότερα features ώστε να μειώσουμε τις παραμέτρους (άρα και τον χρόνο υπολογισμού) κρατώντας την αρχική ακρίβεια ‘ίδια’.

# Features	#parameters	Train. Acc.	Val. Acc.	Test Acc.	Matthews	Calc. time
Χωρίς PCA						
21	792	91.43	82.67	83.54	0.80	143
15	510	87.97	82.55	83.33	0.80	140
10	330	77.96	76.89	73.47	0.68	143
Με PCA						
21	792	95,03	84,43	83,33	0,80	143
15	87,63	75,47	82,86	82,86	0,80	135
10	330	80.24	76.42	78.64	0.74	137

Με όλες τις μεταβλητές βλέπουμε ότι ο αλγόριθμος αποδίδει με ακρίβεια κοντά στο 92%/84% (Train/Test) ενώ με 6 λιγότερες διατηρεί κοντινό test acc. και χάνουμε μόνο στο training. Και οι 2 προσεγγίσεις (PCA και χωρίς) δε δείχνουν μεγάλες διαφορές ωστόσο αν θελήσουμε να μειώσουμε περεταίρω το πλήθος των features βλέπουμε πως η PCA αποδίδει καλύτερα καθώς με 10 features οι διαφορές είναι εμφανείς.