Θεωρία Αποφάσεων
2η Εργασία

Χειμερινό Εξάμηνο 2024 – 2025

31 Ιανουαρίου 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Τμήμα Μηχανικών Η / Υ και Πληροφορικής
Πολυτεχνική Σχολή

Όνομα: Μηλτιάδης
Επώνυμο: Μαντές
Α.Μ.: 1084661
E – mail: up1084661@ac.upatras.gr
Εξάμηνο: 9ο

Διδάσκων: Δημήτριος Κοσμόπουλος
Τομέας Εφαρμογών και Θεμελιώσεων της Επιστήμης των Υπολογιστών

Επιλεγόμενο Μάθημα – CEID_NE5237

ΘΕΜΑ: Παραχάραξη Χαρτονομισμάτων

Περιεχόμενα

0 Εισαγωγή

Το dataset μας περιλαμβάνει 4 γνωρίσματα (**variance, skewness, curtosis, entropy**) και το label **'class'**, το οποίο αναφέρει σε ποια κλάση αντιστοιχεί κάθε εγγραφή του dataset. Πρόκειται στην ουσία για ένα task δυαδικής κατηγοριοποίησης, καθώς χρησιμοποιούνται μόνο 2 κλάσεις, οι 0 και 1.

Data Preprocessing

Προτού χρησιμοποιήσουμε τα δεδομένα του dataset, θα χρειαστεί να τα κανονικοποιήσουμε έτσι ώστε να αποτρέψουμε χαρακτηριστικά με μεγαλύτερες τιμές ή διαφορετική κλίμακα να κυριαρχήσουν στην απόφαση για την ταξινόμηση. Για τον λόγο αυτό χρησιμοποιούμε τον Standard Scaler, ο οποίος μετασχηματίζει τα δεδομένα έτσι ώστε να έχουν μέση τιμή (mean) ίση με 0 και τυπική απόκλιση (standard deviation) ίση με 1. Η διαδικασία που ακολουθεί είναι να υπολογίσει τη μέση τιμή και τη τυπική απόκλιση για κάθε feature του dataset και έπειτα να εφαρμόσει σε αυτή τον ακόλουθο μετασχηματισμό: $z = \frac{x-\mu}{\sigma}$.

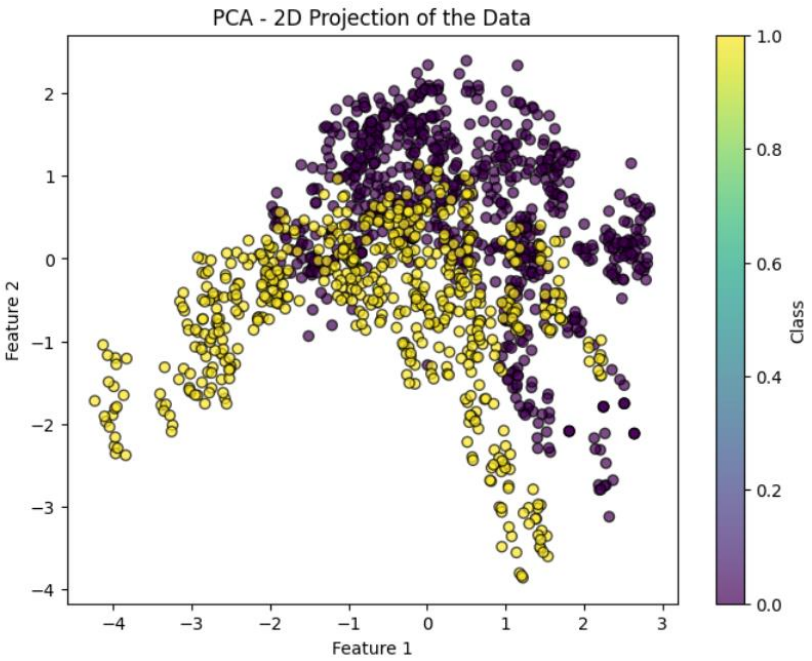
Επίσης, εμφανίζουμε και το πλήθος των δειγμάτων που διαθέτουμε σε κάθε κλάση πριν τις διαχωρίσουμε σε training, validation και testing sets.

```
Sample points for each class before splitting:
Class 0: 762 samples
Class 1: 610 samples
```

1 Ταξινόμηση με Parzen Windows

Ξεκινάμε εφαρμόζοντας **Principal Component Analysis (PCA)** στα κανονικοποιημένα δεδομένα προκειμένου να κάνουμε μείωση της διαστατικότητάς τους. Κάθε principal component feature θα προκύπτει ως γραμμικός συνδυασμός των 4 αρχικών features που διαθέτουμε. Επίσης, για καλύτερη κατανόηση εκτυπώνουμε και την επίδραση κάθε χαρακτηριστικού στην εξαγωγή των δύο τελικών principal components. Για να το πετύχουμε αυτό, χρησιμοποιούμε το αντικείμενο **PCA** της βιβλιοθήκης sklearn που μειώνει τα δεδομένα σε 2 διαστάσεις και μετά μέσω της **fit_transform ( )** τα μετασχηματίζει σε 2-d χώρο.

Από την οπτικοποίηση των δεδομένων παρατηρούμε ότι τα δεδομένα διαχωρίζονται με τέτοιο τρόπο ώστε να **μπορούν να σχηματίσουν δύο διακριτές ομάδες**, δηλαδή μπορούν να διαχωριστούν σε **δύο διαφορετικές κλάσεις**. Ο διαχωρισμός φαίνεται να είναι σχετικά καλός, παρ' όλο που υπάρχει και κάποια επικάλυψη, γεγονός το οποίο υποδηλώνει ότι το μοντέλο ταξινόμησης που θα εφαρμόσουμε θα μπορούσε να επιτύχει ικανοποιητικές επιδόσεις.
Γενικά, τα δεδομένα μοιάζουν να **μην είναι γραμμικώς διαχωρίσιμα** λόγω της επικάλυψης, γεγονός που μπορεί να δυσκολέψει στη συνέχεια έναν Linear Classifier.



Εικόνα 1: PCA data visualization

Επιπλέον, παρατηρούμε ότι στη **πρώτη κύρια συνιστώσα** το μεγαλύτερο βάρος διαθέτουν τα χαρακτηριστικά **skewness, curtosis**, ενώ στη **δεύτερη κύρια συνιστώσα** μεγαλύτερη επίδραση έχουν τα χαρακτηριστικά **variance** και **entropy**. Αυτό μας δείχνει ότι η πληροφορία στα δεδομένα έχει όντως κατανεμηθεί σε διαφορετικές "κατευθύνσεις" (συνιστώσες), όπως είναι άλλωστε και ο στόχος του PCA.

```
Principal Components:
    variance  skewness  curtosis   entropy
PC1  0.248772  0.639323 -0.612707 -0.392389
PC2  0.754591 -0.050340 -0.153459  0.636010
```

Στη συνέχεια, χωρίζουμε τα PCA reduced δεδομένα σε αναλογία **60-20-20** και βλέπουμε πάλι για κάθε κλάση πόσα δείγματα αντιστοιχούν σε κάθε set.

```
Sample points for each class after splitting:

Training Set:
Class 0: 457 samples (59.97% of class 0)
Class 1: 366 samples (60.00% of class 1)
```

```
Validation Set:
Class 0: 152 samples (19.95% of class 0)
Class 1: 122 samples (20.00% of class 1)

Test Set:
Class 0: 153 samples (20.08% of class 0)
Class 1: 122 samples (20.00% of class 1)
```

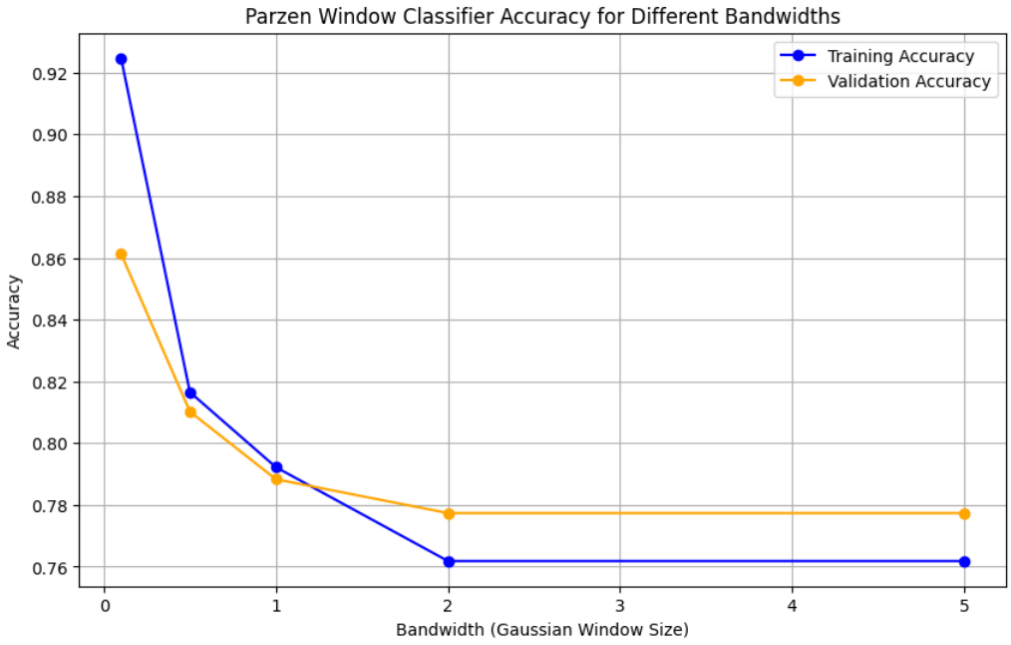
Προχωράμε τώρα στην υλοποίηση του Parzen Window, η οποία πραγματοποιείται στη συνάρτηση **parzen_window( )** με ορίσματα τα **X_train** (δεδομένα εκπαίδευσης), **y_train** (labels εκπαίδευσης), **X_test** (δεδομένα ελέγχου) και **bandwidth** (εύρος ζώνης του Gaussian kernel). Αποθηκεύουμε αρχικά τις διακριτές κλάσεις των labels εκπαίδευσης στην μεταβλητή classes μέσω της **unique( )** και έπειτα ορίζουμε τη λίστα **predictions**, όπου θα αποθηκεύονται οι προβλέψεις για τα δεδομένα εκπαίδευσης. Επίσης, ορίζουμε και το dictionary **pdfs**, το οποίο θα χρησιμοποιήσουμε μετά για να υπολογίσουμε τις εκτιμήσεις της πυκνότητας πιθανότητας (PDF) για κάθε κλάση για όλα τα δείγματα δοκιμής που έχουν αποθηκευτεί στη λίστα **predictions**. Στη συνέχεια, για κάθε κλάση **c** κρατάμε μόνο τα δεδομένα εκπαίδευσης που αντιστοιχούν σε αυτήν και τα αποθηκεύουμε στη **X_class** με την εντολή **X_class = X_train[y_train == c]**. Χρησιμοποιούμε αυτά τα δεδομένα για να υπολογίσουμε τη πυκνότητα της εκάστοτε κλάσης, η οποία αποθηκεύεται στη μεταβλητή **density**, την οποία στη συνέχεια κανονικοποιούμε διαιρώντας με το πλήθος των στοιχείων της κλάσης (**len(X_class)**). Εφόσον χρησιμοποιούμε Gaussian kernel, ο υπολογισμός της πυκνότητας θα δίνεται από τον εξής τύπο:

$$K(x) = \exp\left(-\frac{\|x_{\text{test}} - x_{\text{train}}\|^2}{2 \cdot \text{bandwidth}^2}\right)$$

Η υπολογισμένη πυκνότητα για την κλάση **c** προστίθεται στη λίστα **class_probabilities**. Από τη λίστα αυτή μέσω της **argmax( )** υπολογίζουμε ποια κλάση έχει τη μεγαλύτερη πιθανότητα και αυτή θα επιλεγεί ως η πρόβλεψη για τη ταξινόμηση κάθε δείγματος από το **X_test**. Παράλληλα, η εκτιμώμενη πυκνότητα για την κλάση **c** αποθηκεύεται και στη λίστα **pdfs[c]**, που αντιστοιχεί επίσης στο τρέχον δείγμα δοκιμής **X_test**.

Καλούμε τη συνάρτηση αυτή για διαφορετικές τιμές του **bandwidth**, έτσι ώστε να καταλήξουμε ποια τιμή είναι η πιο κατάλληλη για να χρησιμοποιήσουμε στο Parzen Window μας.

Εικόνα 2: Accuracy for different h values



Παρατηρούμε λοιπόν ότι η ακρίβεια εκπαίδευσης (Training Accuracy) είναι υψηλή για μικρές τιμές του bandwidth, αλλά μειώνεται απότομα καθώς το bandwidth αυξάνεται. Ακόμη, η ακρίβεια επικύρωσης (Validation Accuracy) φτάνει τη μέγιστη τιμή της σε χαμηλές έως μέτριες τιμές του bandwidth (γύρω στο 0.5-1.0) και έπειτα σταθεροποιείται για μεγαλύτερες τιμές.

Αυτό υποδηλώνει ότι **χαμηλές τιμές bandwidth** οδηγούν σε υπερεκπαίδευση (**overfitting**), ενώ πολύ **μεγάλες τιμές** οδηγούν σε υποεκπαίδευση (**underfitting**).

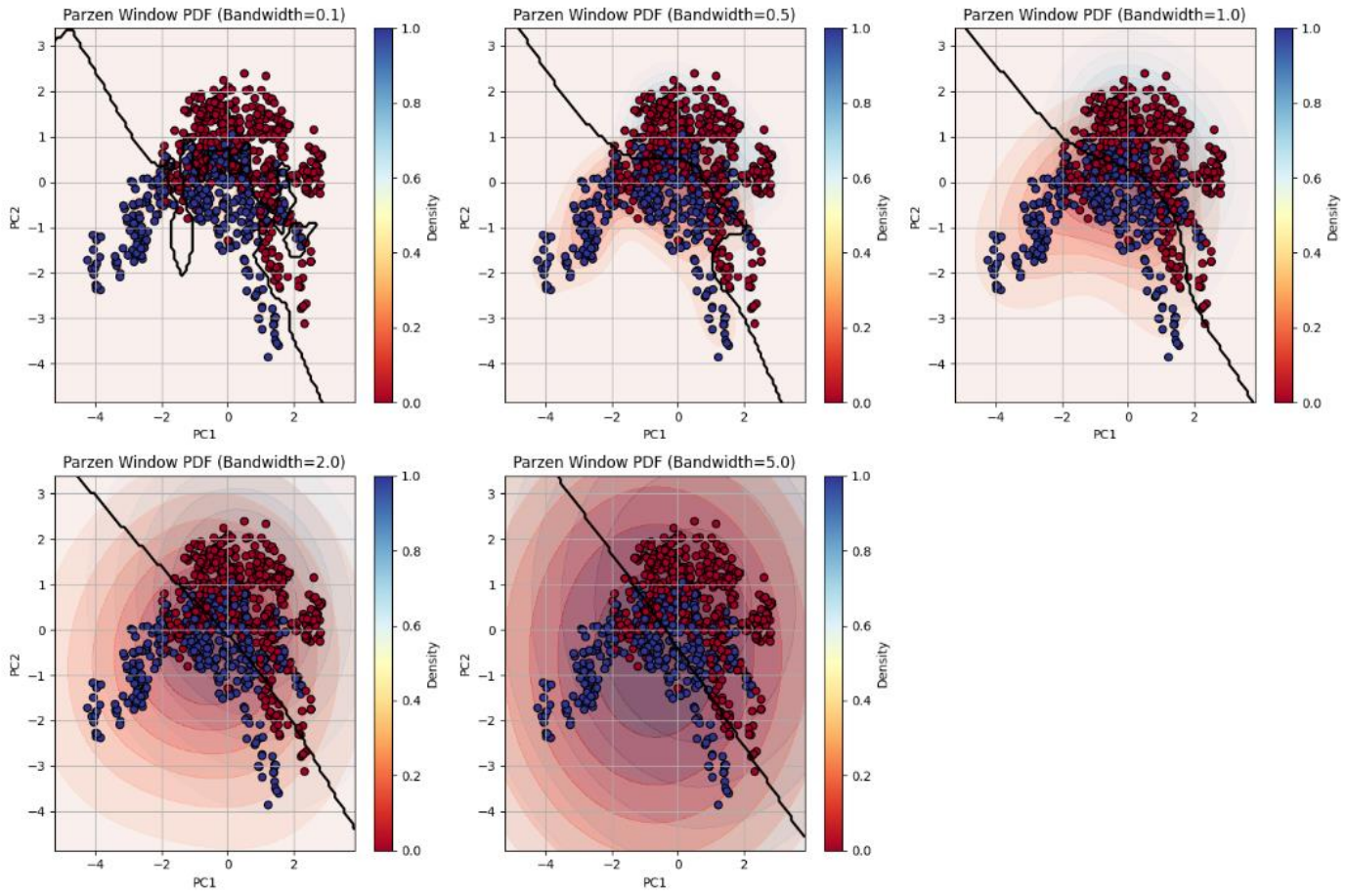
Σημειώνουμε ότι εφόσον πρόκειται για πρόβλημα ταξινόμησης σχεδιάζουμε τη γραφική με βάση το accuracy και όχι το MSE ή παρεμφερείς μετρικές, οι οποίες προτιμώνται συνήθως σε προβλήματα παλινδρόμησης που χειρίζονται αριθμητικά δεδομένα.

Επιπλέον, σχεδιάζουμε την PDF που προκύπτει για κάθε τιμή του bandwidth μαζί με το όριο απόφασης, προκειμένου να δούμε πιο καθαρά πώς μεταβάλλεται ο διαχωρισμός των δύο κλάσεων όσο μεταβάλλεται και η PDF.

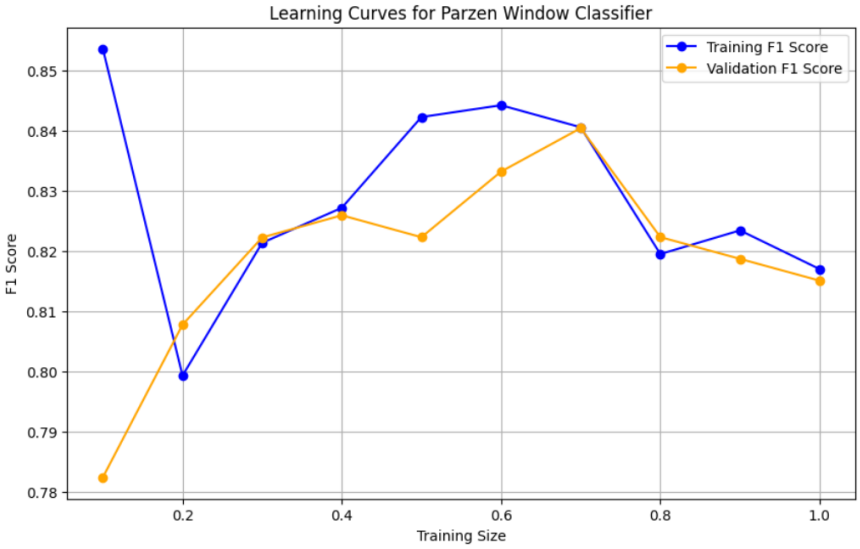
Όσον αφορά την κατανομή της πυκνότητας πιθανότητας, όμοια με πριν βλέπουμε ότι για πολύ **μικρό bandwidth** η κατανομή των δεδομένων είναι πολύ λεπτομερής, με αποτέλεσμα να προσπαθεί το μοντέλο να ταιριάξει υπερβολικά στα δεδομένα (**overfitting**). Αυτό επιβεβαιώνεται και από **το όριο απόφασης**, το οποίο είναι **εξαιρετικά περίπλοκο** και **ακολουθεί πολύ στενά την κατανομή των σημείων** των δεδομένων. Για **ενδιάμεσες τιμές** οι κατανομές δείχνουν μια **ισορροπημένη περιγραφή** των δεδομένων, με καλό διαχωρισμό, ενώ **το όριο απόφασης δείχνει και αυτό να έχει εξομαλυνθεί** αρκετά. Τέλος, για **μεγάλες τιμές** bandwidth η κατανομή **εξομαλύνεται υπερβολικά**, χάνοντας την ικανότητά της να διαχωρίσει τις κλάσεις σωστά (**underfitting**). Το **όριο απόφασης** είναι υπερβολικά απλοποιημένο, **σχεδόν γραμμικό**, και δεν μπορεί να συλλάβει τη δομή των δεδομένων.

Συμπεραίνουμε επομένως ότι το καλύτερο trade-off επιτυγχάνεται όταν η τιμή του bandwidth κυμαίνεται στο διάστημα **0.5 – 1**. **Για την υλοποίηση του Parzen Window θα επιλέξουμε να χρησιμοποιήσουμε h=0.5.**

Εικόνα 3: PDFs and decision boundaries for different h values

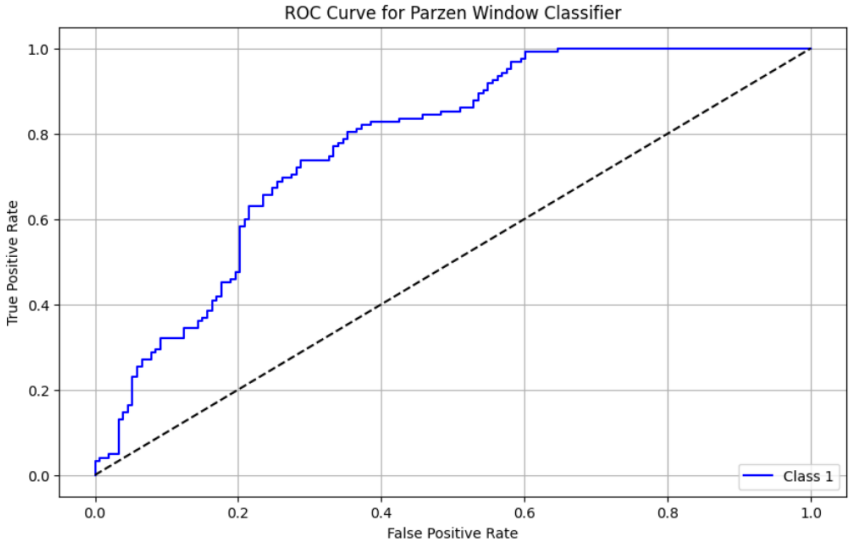


Αφού υλοποιήσουμε το συγκεκριμένο Parzen Window, σχεδιάζουμε τα learning και ROC curves, προκειμένου να κάνουμε ένα καλό evaluation του μοντέλου, σε συνδυασμό με τη μετρική F1-Score. Τα learning curves μας δείχνουν πώς αλλάζει η απόδοση (F1-Score) του μοντέλου με βάση το μέγεθος του συνόλου εκπαίδευσης και δημιουργούνται στη συνάρτηση **generate_learning_curves( )**. Πρώτα δημιουργείται μια λίστα με τιμές από **10% έως 100%** του συνόλου εκπαίδευσης, χρησιμοποιώντας την εντολή **np.linspace(0.1, 1.0, 10)**. Αυτές οι τιμές αντιστοιχούν στα ποσοστά του συνόλου εκπαίδευσης που θα χρησιμοποιηθούν. Για κάθε ποσοστό του συνόλου εκπαίδευσης υπολογίζουμε τις προβλέψεις για το υποσύνολο εκπαίδευσης (**y_train_pred**) και το πλήρες σύνολο δοκιμής (**y_val_pred**) και έπειτα τις χρησιμοποιούμε για να εκτιμήσουμε τα αντίστοιχα F1-Scores. Για το ROC curve χρησιμοποιούμε την αντίστοιχη PDF που περιέχει τις πιθανότητες που αντιστοιχούν σε κάθε κλάση για τα δείγματα του **X_test.** Η γραμμή **y_prob = np.array(pdfs[1])** εξάγει τις πιθανότητες που αντιστοιχούν στην **κλάση 1**, ώστε να χρησιμοποιηθούν στον υπολογισμό του ROC και της AUC. Τέλος, μετατρέπουμε τις πραγματικές ετικέτες του **y_test** σε δυαδική μορφή και υπολογίζουμε ROC και AUC για την κλάση 1. Η συνάρτηση **roc_curve( )** επιστρέφει τα **tpr**, **fpr** (True-Positive Rate, False-Positive Rate) και η συνάρτηση **auc( )** τα δέχεται στη συνέχεια σαν ορίσματα για να υπολογίσει το εμβαδόν κάτω από τη καμπύλη. Τέλος, υπολογίζουμε και το F1-Score για όλο το testing set και παρακάτω βλέπουμε τα αποτελέσματα:



Εικόνα 4: Learning curve for training and validation sets

Για πολύ μικρά μεγέθη εκπαίδευσης (training size < 20%), το F1 score στο σύνολο εκπαίδευσης είναι υψηλό, ενώ στο σύνολο επικύρωσης είναι πιο χαμηλό. Με την αύξηση του μεγέθους του συνόλου εκπαίδευσης, οι καμπύλες εκπαίδευσης και επικύρωσης συγκλίνουν. Περίπου στο **training size = 60%–80%**, το F1 score φτάνει την κορυφή. Αυτή η σύγκλιση υποδεικνύει ότι το μοντέλο **μαθαίνει καλύτερα με περισσότερα δεδομένα και γενικεύει καλά**.



Εικόνα 5: ROC curve for class 1

Η καμπύλη ROC βρίσκεται σαφώς πάνω από τη διαγώνιο που αντιπροσωπεύει έναν τυχαίο ταξινομητή. Αυτό δείχνει ότι ο ταξινομητής Parzen Window αποδίδει καλύτερα από τυχαία πρόβλεψη. Επίσης, στα χαμηλά επίπεδα του False Positive Rate (FPR), το True Positive Rate (TPR) αυξάνεται γρήγορα, κάτι που δείχνει ότι **ο ταξινομητής έχει καλή ευαισθησία (sensitivity) για αρκετά thresholds και άρα αποδίδει καλά σε χαμηλά false positive rates**.

Average AUC Parzen Window: 0.7746

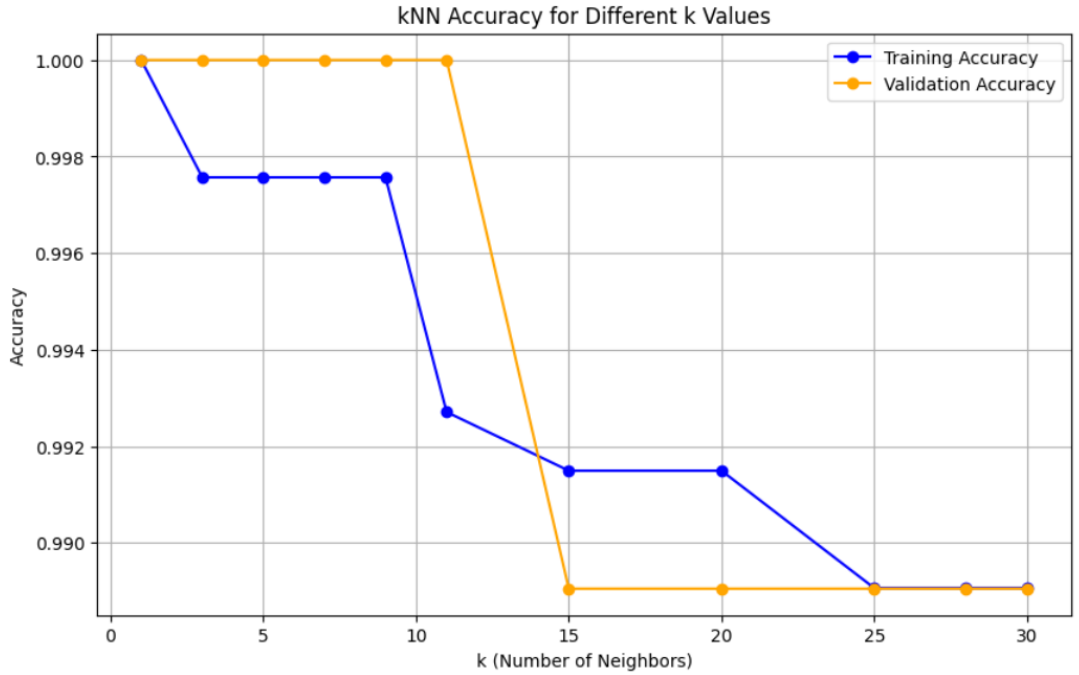
Το AUC δείχνει ότι ο ταξινομητής έχει **ικανοποιητική συνολική ικανότητα διάκρισης** μεταξύ της θετικής (κλάση 1) και της αρνητικής κλάσης (κλάση 0). Πρακτικά, αυτό σημαίνει ότι κατά μέσο όρο, ο ταξινομητής έχει **77.46% πιθανότητα να κατατάξει σωστά ένα τυχαίο θετικό δείγμα υψηλότερα από ένα τυχαίο αρνητικό δείγμα**.

F1 Score for Parzen Window: 0.8151

Ένα υψηλό F1-Score σημαίνει ότι το μοντέλο είναι αποτελεσματικό τόσο στο να εντοπίζει σωστά τα θετικά παραδείγματα (καλό recall) όσο και στο να αποφεύγει λάθη σε αρνητικά παραδείγματα (καλό precision). Στη συγκεκριμένη περίπτωση, φαίνεται ότι **ο ταξινομητής έχει καλή ισορροπία μεταξύ ακρίβειας και ανάκλησης**. **Το υψηλό F1 Score συνδυαζόμενο με το AUC υποδεικνύει ότι το μοντέλο είναι σχετικά καλό τόσο στην πρόβλεψη όσο και στη διάκριση των κατηγοριών.**

2 Ταξινόμηση με k-NN

Τώρα χωρίζουμε τα αρχικά κανονικοποιημένα δεδομένα χωρίς PCA πάλι σε αναλογία **60-20-20** και εξετάζουμε την απόδοση του ταξινομητή για τις διάφορες τιμές της παραμέτρου k, δηλαδή του πλήθους των πλησιέστερων γειτόνων του δείγματος. Θα χρησιμοποιήσουμε το αντικείμενο **kNeighborsClassifier** της βιβλιοθήκης sklearn, το οποίο το εκπαιδεύουμε μέσω της συνάρτησης **fit ( )** πάνω στα **X_train**, **y_train**. Η πρόβλεψη για κάθε δείγμα γίνεται επίσης μέσω της συνάρτησης **predict( )**. Υπολογίζουμε τα διάφορα accuracies για ένα εύρος k από 1-30 και παρατηρούμε το εξής:

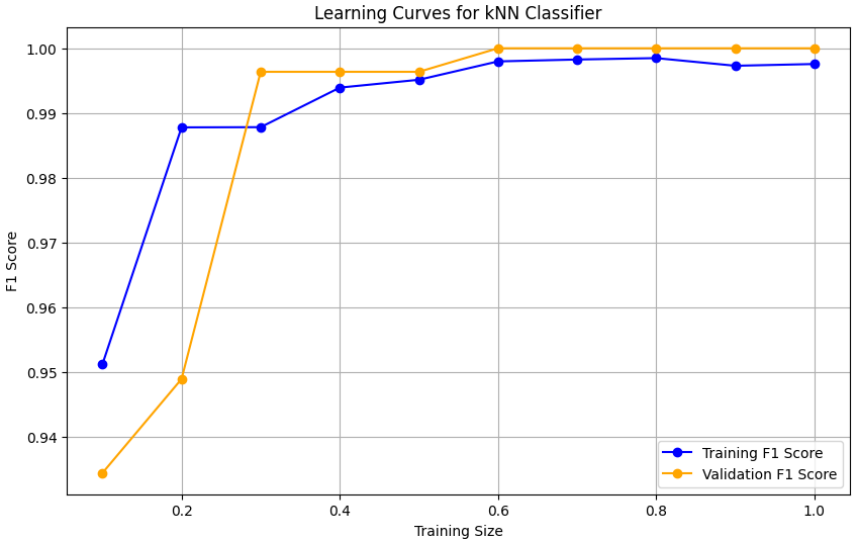


Εικόνα 6: Accuracy for different k values

Για **μικρές τιμές του k (1 έως 11)** η **ακρίβεια εκπαίδευσης** είναι πολύ υψηλή (**σχεδόν 1**), το οποίο σημαίνει ότι το μοντέλο προσαρμόζεται πολύ καλά στα δεδομένα εκπαίδευσης. Η **ακρίβεια στην επικύρωση** είναι επίσης **σταθερά στο 1**, που σημαίνει ότι το μοντέλο **γενικεύει πολύ καλά και στις μη γνωστές παρατηρήσεις**. Από **k = 15 και πάνω**, η **ακρίβεια εκπαίδευσης μειώνεται** περισσότερο, και η **ακρίβεια επικύρωσης** παραμένει **σταθερή**, οπότε φαίνεται ότι το μοντέλο έχει φτάσει σε μια "κορεσμένη" κατάσταση, όπου δεν υπάρχει περαιτέρω βελτίωση. Για πολύ μικρές τιμές ωστόσο του k υπάρχει ο φόβος μην το μοντέλο εκπαιδευτεί στο να δείχνει μεγάλη ευαισθησία σε μικρές διακυμάνσεις των δεδομένων. Από την άλλη, αν χρησιμοποιούμε πολλούς γείτονες για τη ταξινόμηση το μοντέλο μπορεί να μάθει να αγνοεί τις τοπικές ιδιαιτερότητες των δεδομένων. Συνεπώς, θέλουμε να επιλέξουμε μια τιμή του k η οποία θα μας προσφέρει καλή γενίκευση και σταθερότητα και να αποφεύγονται όλοι οι παραπάνω κίνδυνοι. Συνεπώς, εφόσον δεν υπάρχει εμφανής υποπροσαρμογή όσο αυξάνεται το k, μια καλή επιλογή για το k θα ήταν από 5-10 για μεγαλύτερη ασφάλεια. **Για την υλοποίηση λοιπόν του k-NN classifier θα επιλέξουμε να χρησιμοποιήσουμε k=5.**

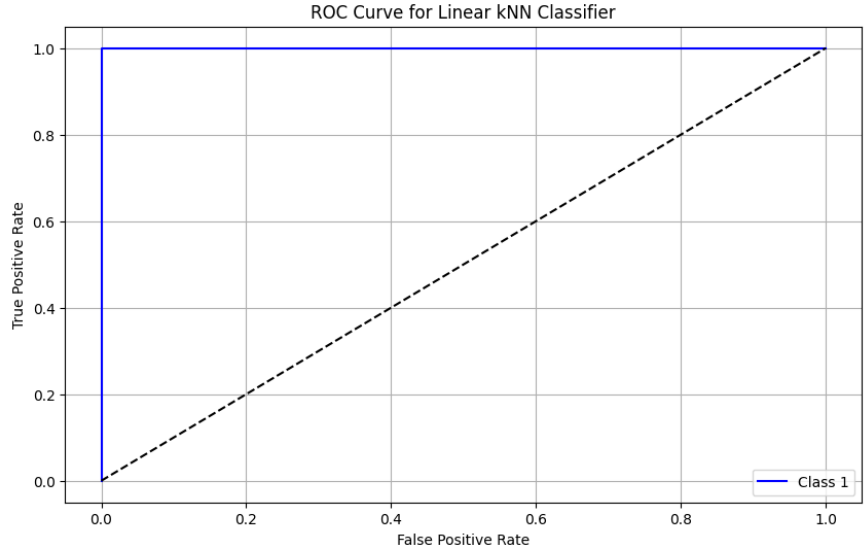
Αφού υλοποιήσουμε τον συγκεκριμένο kNN Classifier, κατασκευάζουμε πάλι με το ίδιο σκεπτικό την συνάρτηση **generate_learning_curves( )** για να εκτυπώσουμε τα learning curves του training και validation set. Για τη δημιουργία του ROC curve χρησιμοποιούμε την εντολή **y_prob = knn.predict_proba(X_test)[:, 1]**, η οποία εξάγει τις πιθανότητες για τη θετική κατηγορία (class 1) και στη συνέχεια πάλι θέτουμε στη συνάρτηση **roc_curve( )** τα **y_proba**

και **y_test**, έτσι ώστε να επιστραφούν τα tpr και fpr. Τέλος, τα AUC και F1-Score υπολογίζονται και αυτά με το ίδιο σκεπτικό με πριν. Παρακάτω βλέπουμε τα αποτελέσματα:



Εικόνα 7: Learning curve for training and validation set

Παρατηρούμε ότι το μοντέλο χρειάζεται ένα αρκετά μεγάλο ποσοστό δεδομένων για να φτάσει τη μέγιστη απόδοσή του, αλλά ακόμα και με λιγότερα δεδομένα (περίπου 50%), η απόδοση είναι ήδη πολύ καλή. Η χρήση περισσότερων δεδομένων εκπαίδευσης βελτιώνει σταδιακά την απόδοση, αλλά **το μοντέλο φτάνει τη μέγιστη απόδοση του αρκετά νωρίς (στο 70-80% των δεδομένων)**. Εν τέλει, **οι δύο καμπύλες συγκλίνουν, άρα η απόδοσή του μοντέλου στα δεδομένα επικύρωσης είναι σχεδόν ίδια με αυτή στα δεδομένα εκπαίδευσης.**



Εικόνα 8: ROC curve for class 1

Η καμπύλη ROC είναι σχεδόν τέλεια και πλησιάζει το πάνω αριστερό μέρος του διαγράμματος, το οποίο σημαίνει ότι **ο ταξινομητής έχει εξαιρετική απόδοση**. Άρα, ο ταξινομητής **διαχωρίζει τις δύο κατηγορίες σχεδόν τέλεια** και η πιθανότητα να κάνει λάθος (False Positive ή False Negative) είναι εξαιρετικά χαμηλή, αφού δεν μπερδεύει τα αρνητικά δείγματα ως θετικά.


```
Average AUC for kNN: 1.0000
```

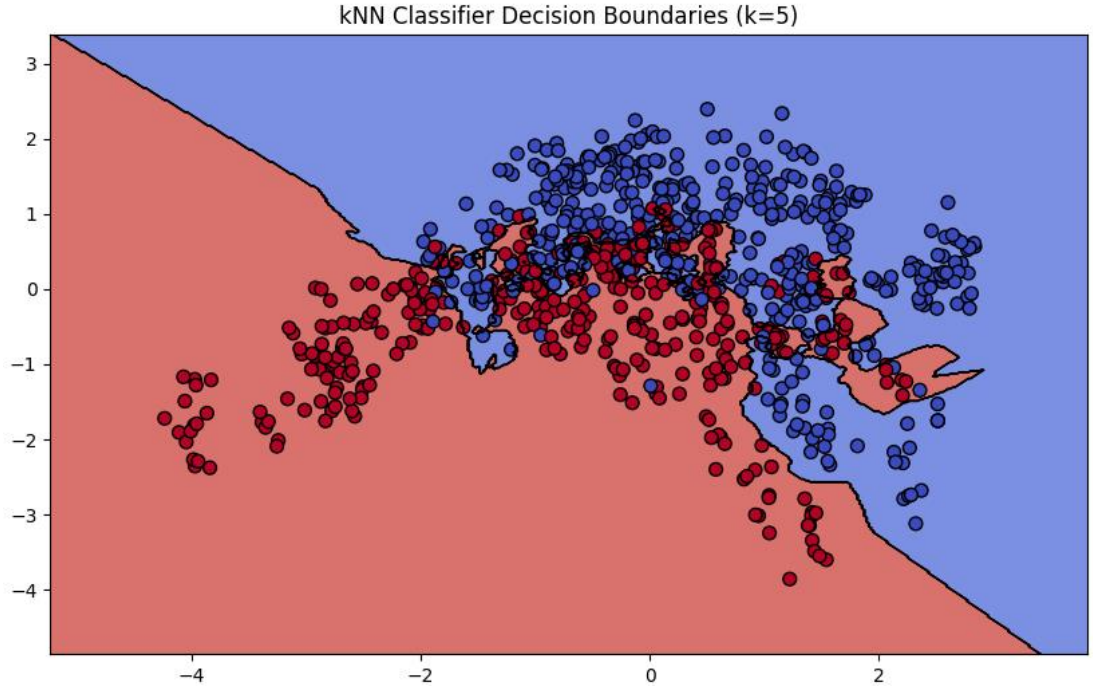
Το AUC δείχνει ότι ο ταξινομητής έχει **τελεία συνολική ικανότητα διάκρισης** μεταξύ της θετικής (κλάση 1) και της αρνητικής κλάσης (κλάση 0).

```
F1 Score for k-NN: 1.0000
```

Η τιμή του F1-Score είναι μια ακόμα ένδειξη ότι ο ταξινομητής έχει **τέλεια απόδοση**, καθώς επιτυγχάνει απόλυτη ακρίβεια και ανάκληση.

Τέλος, σχεδιάζουμε τα όρια απόφασης του ταξινομητή με τη συνάρτηση **plot_knn_decision_boundaries( )** απεικονίζοντας πάλι τα δεδομένα μας στις δύο διαστάσεις. Βλέπουμε ότι ο ταξινομητής k-NN με k=5 δημιουργεί **όρια απόφασης που προσαρμόζονται αρκετά στα δεδομένα**. Πιο συγκεκριμένα, σε περιοχές όπου οι δύο κλάσεις είναι κοντά (π.χ.

στο κέντρο) τα όρια απόφασης είναι πιο περίπλοκα. Αυτό δείχνει ότι ο k-NN μπορεί να **χειριστεί καλά περιοχές όπου οι κλάσεις δεν είναι γραμμικά διαχωρίσιμες**.



Εικόνα 9: Decision boundaries for k-NN classifier

3 Ταξινόμηση με Linear SVM

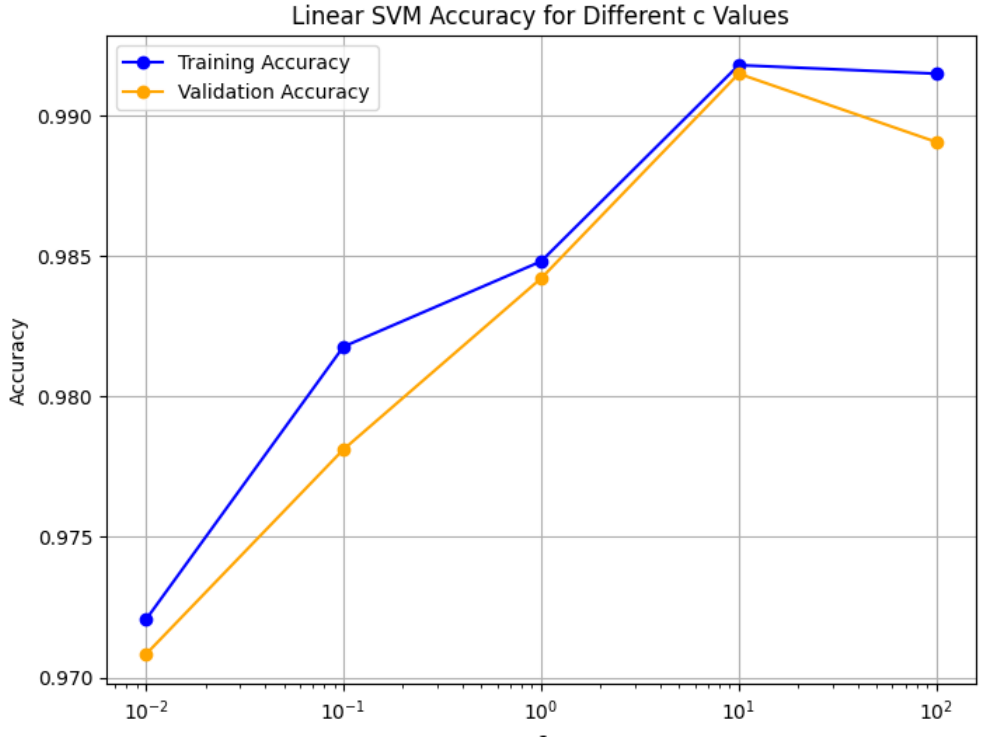
Πάλι χωρίζουμε τα αρχικά κανονικοποιημένα δεδομένα χωρίς PCA πάλι σε αναλογία **60-20-20** και εξετάζουμε την απόδοση του ταξινομητή για τις διάφορες τιμές της υπερπαραμέτρου c, η οποία ελέγχει το trade-off μεταξύ της ελαχιστοποίησης του σφάλματος ταξινόμησης και της μεγιστοποίησης του περιθωρίου (margin) για κάθε δείγμα. Θα χρησιμοποιήσουμε το αντικείμενο **SVC** της βιβλιοθήκης sklearn με όρισμα **kernel='linear'** για γραμμικό πυρήνα, το οποίο το εκπαιδεύουμε μέσω της συνάρτησης **fit ( )** πάνω στα **X_train**, **y_train**. Η πρόβλεψη για κάθε δείγμα γίνεται επίσης μέσω της συνάρτησης **predict( )**. Ωστόσο, στη προκειμένη περίπτωση θα αλλάξουμε τρόπο προσέγγισης για την εύρεση της βέλτιστης υπερπαραμέτρου πριν την εφαρμογή του μοντέλου, καθώς θα εφαρμόσουμε **5-fold cross validation** για ένα εύρος c από 0.01-100 χρησιμοποιώντας το αντικείμενο **GridSearchCV( )**. Αν κάνουμε **fit( )** αυτό το αντικείμενο πάνω στα δεδομένα εκπαίδευσης και καλέσουμε πάνω σε αυτό την εντολή **best_params_** λαμβάνουμε το εξής:

```
Best Linear SVM Parameters: {'C': 10}
```

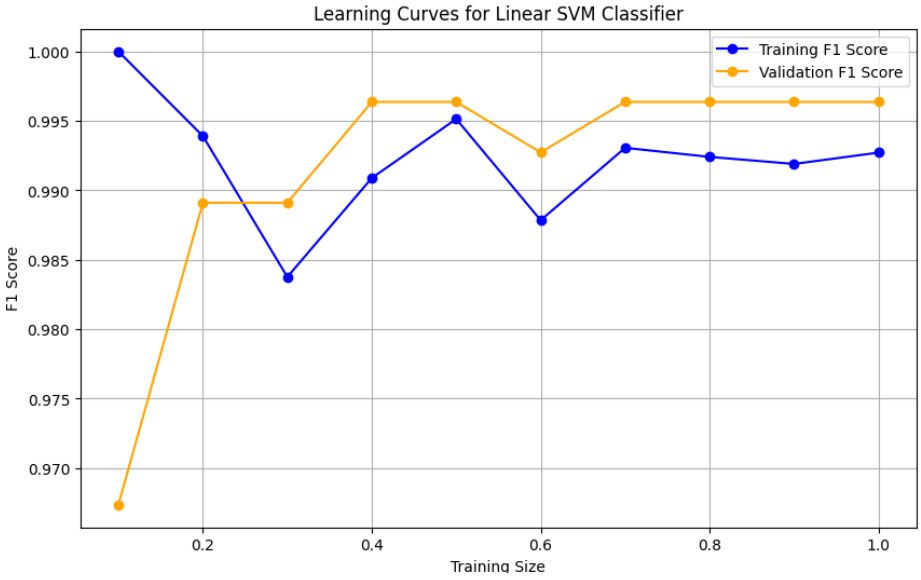
Μια τέτοια τιμή του c υποδηλώνει ένα σχετικά ισχυρό regularization, με αποτέλεσμα το μοντέλο να «τιμωρεί» λιγότερο τα λιγότερο σύνθετα όρια απόφασης. Έτσι, το μοντέλο δεν θα είναι υπερβολικά ευαίσθητο σε συγκεκριμένα σημεία του dataset, ενώ παράλληλα θα μπορεί να γενικεύει πολύ καλά σε άγνωστα δεδομένα ελέγχου.

Επίσης, σχεδιάζουμε τα διάφορα accuracies για το εύρος c σε λογαριθμική κλίμακα, έτσι ώστε να είναι πιο εμφανής η επίδραση του c. Παρατηρούμε πως για **μικρές τιμές** του c τόσο η **ακρίβεια εκπαίδευσης** όσο **και η ακρίβεια επικύρωσης** ξεκινάνε σε **χαμηλότερες τιμές** και όσο **το c αυξάνεται** συγκλίνουν και **σταθεροποιούνται** πολύ κοντά στο 0.99. Αυτό σημαίνει ότι από εκείνο το σημείο και μετά η επίδραση του c να είναι τόσο έντονη, καθώς το μοντέλο έχει εκπαιδευτεί καλά στα υπάρχοντα δεδομένα και έχει μάθει να γενικεύει καλά και σε καινούργια. **Άρα, θα επιλέξουμε να χρησιμοποιήσουμε για την υλοποίηση του Linear SVM τη τιμή c=10**.

Εικόνα 10: Accuracy for different c values

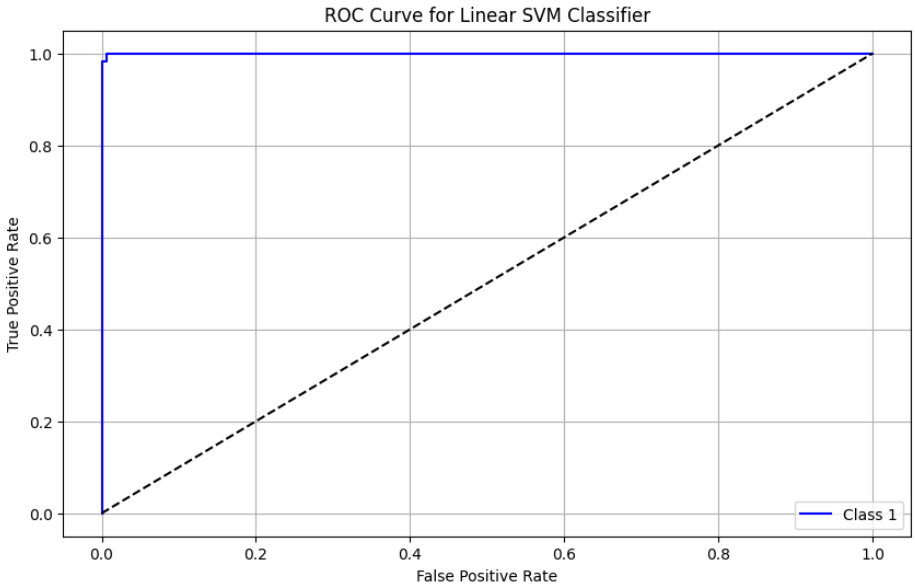


Αφού υλοποιήσουμε τον συγκεκριμένο Linear SVM Classifier, κατασκευάζουμε πάλι με το ίδιο σκεπτικό την συνάρτηση **generate_learning_curves( )** για να εκτυπώσουμε τα learning curves του training και validation set. Για τη δημιουργία του ROC curve χρησιμοποιούμε την εντολή **y_prob = svm.predict_proba(X_test)[:, 1],** η οποία εξάγει τις πιθανότητες για τη θετική κατηγορία (class 1) και στη συνέχεια πάλι θέτουμε στη συνάρτηση **roc_curve( )** τα **y_proba** και **y_test**, έτσι ώστε να επιστραφούν τα tpr και fpr. Τέλος, τα AUC και F1-Score υπολογίζονται και αυτά με το ίδιο σκεπτικό με πριν. Παρακάτω βλέπουμε τα αποτελέσματα:



Εικόνα 12: Learning curve for training and validation set

Παρατηρούμε ότι για **μικρό μέγεθος** training set, το **F1-score είναι πολύ υψηλό**. Αυτό είναι αναμενόμενο, καθώς το μοντέλο έχει πολύ λίγα δεδομένα να προσαρμοστεί και μπορεί να τα ταξινομήσει τέλεια. Καθώς αυξάνεται το μέγεθος υπάρχουν κάποιες **μικρές διακυμάνσεις στο validation set**, κυρίως όταν το μέγεθος του training set είναι μεσαίο (**30%-70%**). Τέλος, **οι καμπύλες του training και του validation set συγκλίνουν** καθώς αυξάνεται το μέγεθος του training set. **Αυτό δείχνει ότι το μοντέλο δεν έχει πρόβλημα overfitting ή underfitting.**



Εικόνα 13: ROC curve for class 1

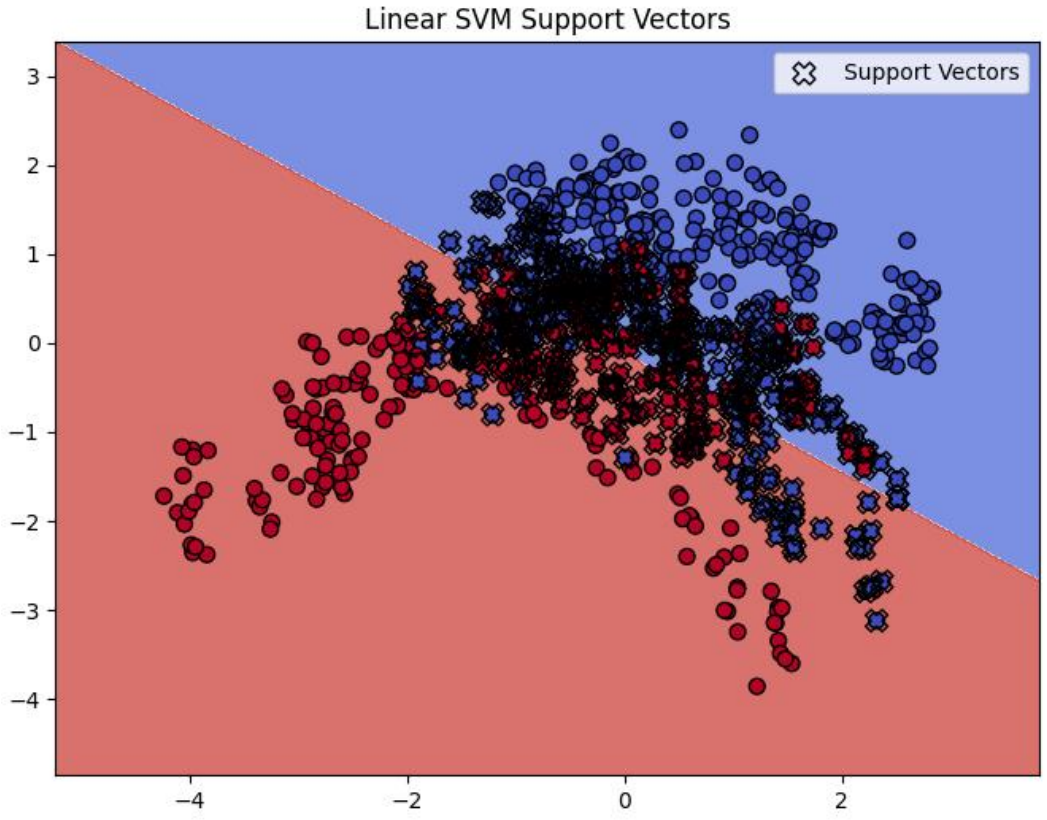
Η ROC καμπύλη είναι πάλι πολύ κοντά στο πάνω αριστερό μέρος, κάτι που υποδεικνύει ότι ο Linear SVM έχει **σχεδόν τέλεια ταξινόμηση**. Πάλι οι δύο κλάσεις διαχωρίζονται σωστά χωρίς να υπάρχει κάποιο μπέρδεμα. Η καλή απόδοση στην ROC καμπύλη μαζί με την απόδοση στις learning curves, υποδεικνύουν ότι το Linear SVM έχει μάθει καλά τα μοτίβα των δεδομένων χωρίς να υπερπροσαρμόζεται.

Average AUC for Linear SVM: 0.9999

Επιβεβαιώνεται πάλι ότι το Linear SVM έχει σχεδόν τέλεια διαχωριστική ικανότητα μεταξύ των δύο κλάσεων.

F1 Score for Linear SVM: 0.9964

Όμοια με πριν, το μοντέλο έχει πολύ καλή απόδοση στην ταξινόμηση, διατηρώντας ισορροπία μεταξύ της ακρίβειας και της ανάκλησης.
Επιπλέον, με τη συνάρτηση **plot_support_vectors( )** επιχειρούμε να οπτικοποιήσουμε τα διανύσματα υποστήριξης στο επίπεδο, τα οποία συγκρατούν το margin από το "πλάτος" του περιθωρίου (margin), δηλαδή την απόσταση μεταξύ της υπερεπιφάνειας διαχωρισμού και των πλησιέστερων σημείων. Από την εικόνα επιβεβαιώνεται ότι **η γραμμή διαχωρισμού που σχηματίζεται από τα διανύσματα υποστήριξης είναι ευθεία**, όπως αναμένεται για το Linear SVM. Ο διαχωρισμός φαίνεται να είναι καλός, με ελάχιστα σημεία από την κόκκινη περιοχή να βρίσκονται στη μπλε περιοχή και αντίστροφα. Συνεπώς, το Linear SVM **χρησιμοποιεί τα πιο κρίσιμα δεδομένα (Support Vectors) για να διαχωρίσει αποτελεσματικά τις δύο κλάσεις**.



Εικόνα 14: Support vectors for Linear SVM

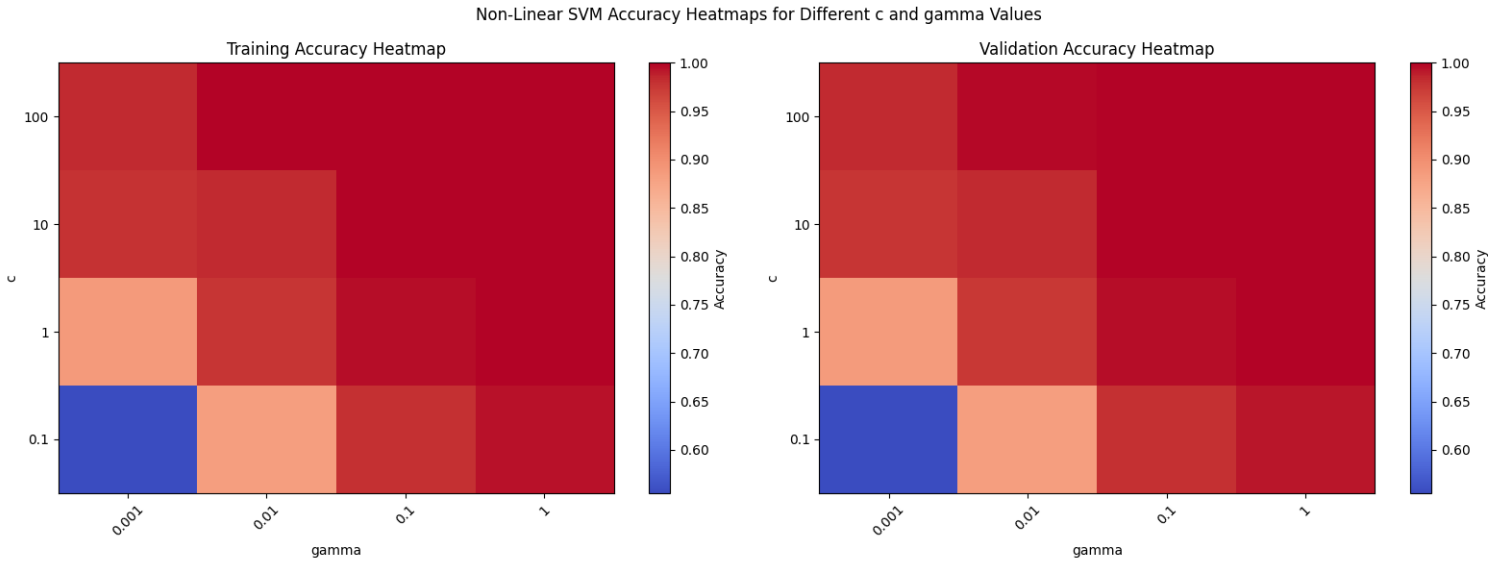
4 Ταξινόμηση με Non-Linear SVM

Πάλι χωρίζουμε τα αρχικά κανονικοποιημένα δεδομένα χωρίς PCA πάλι σε αναλογία **60-20-20**. Εξετάζουμε την απόδοση του ταξινομητή για των κατάλληλο συνδυασμό ανάμεσα στο regularization της υπερπαραμέτρου c και τη λεπτομέρεια της υπερεπιφάνειας διαχωρισμού από την υπερπαράμετρο γ. Στην ουσία το c ρυθμίζει πόσο αυστηρά θέλουμε το SVM να προσπαθεί να ταξινομήσει σωστά τα σημεία, ενώ το γ καθορίζει την επίδραση που έχει κάθε μεμονωμένο σημείο δεδομένων στη διαμόρφωση της υπερεπιφάνειας διαχωρισμού. Θα χρησιμοποιήσουμε το αντικείμενο **SVC** της βιβλιοθήκης sklearn με όρισμα αυτή τη φορά **kernel='rbf'** για μη-γραμμικό πυρήνα, ο οποίος υπολογίζει την ομοιότητα μεταξύ δύο σημείων με βάση την ευκλείδεια απόστασή τους. Το εκπαιδεύουμε μέσω της συνάρτησης **fit ( )** πάνω στα **X_train**, **y_train** και η πρόβλεψη για κάθε δείγμα γίνεται επίσης μέσω της συνάρτησης **predict( )**. Θα χρησιμοποιήσουμε πάλι **5-fold cross validation** για ένα εύρος c από 0.1-100 και γ από 0.001-1 χρησιμοποιώντας το αντικείμενο **GridSearchCV( )**. Αν κάνουμε **fit( )** αυτό το αντικείμενο πάνω στα δεδομένα εκπαίδευσης και καλέσουμε πάνω σε αυτό την εντολή **best_params_** λαμβάνουμε το εξής:

```
Best Non-Linear SVM Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

Η σημασία του c=10 έχει αναλυθεί και στην προηγούμενη ενότητα. Επίσης, μια μεγάλη τιμή γ όπως το 10 πρακτικά σημαίνει ότι **το μοντέλο δίνει μεγάλη έμφαση στα σημεία που βρίσκονται πολύ κοντά μεταξύ τους**, αγνοώντας την επιρροή των πιο μακρινών σημείων. Άρα, το μοντέλο είναι εξαιρετικά τοπικό στις αποφάσεις του, εστιάζοντας σε πολύ μικρές περιοχές του χώρου χαρακτηριστικών. Αν και αυτό μπορεί να αποδώσει πολύ καλά στα δεδομένα εκπαίδευσης, υπάρχει υψηλός κίνδυνος το μοντέλο να μην γενικεύσει καλά σε νέα δεδομένα. Για να αξιολογηθεί η καταλληλότητα αυτής της τιμής, θα πρέπει να γίνει σύγκριση της απόδοσης του μοντέλου στα δεδομένα εκπαίδευσης και επικύρωσης όπως θα δούμε παρακάτω.

Εικόνα 15: Accuracy for different values of c and gamma

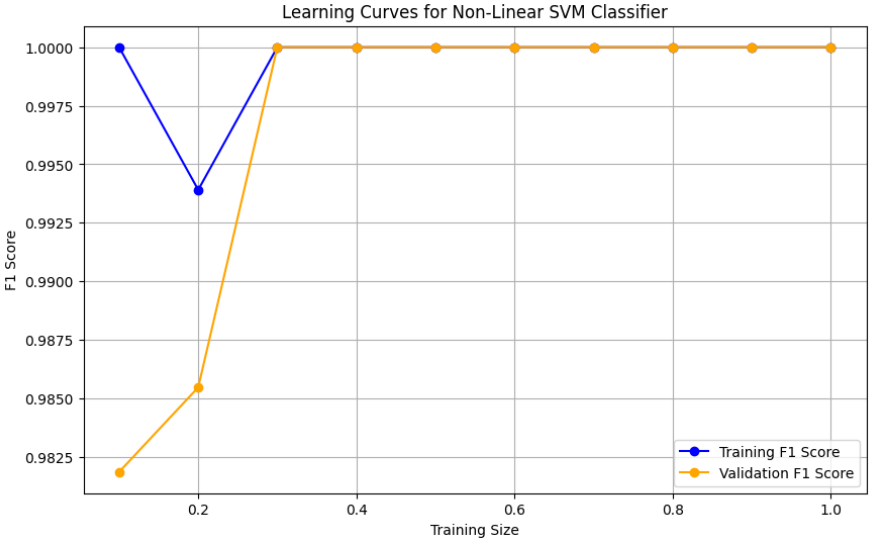


Στο heatmap για την **εκπαίδευση**, παρατηρούμε ότι οι πολύ **μεγάλες τιμές γ και c** επιτυγχάνουν σχεδόν **τέλεια απόδοση**, κάτι που ίσως να δείχνει overfitting. Ωστόσο, στο heatmap της **επικύρωσης**, φαίνεται ότι **οι περιοχές υψηλής ακρίβειας είναι πιο συγκρατημένες**, κάτι που υποδηλώνει **καλύτερη γενίκευση όταν αποφεύγουμε ακραίες τιμές.**

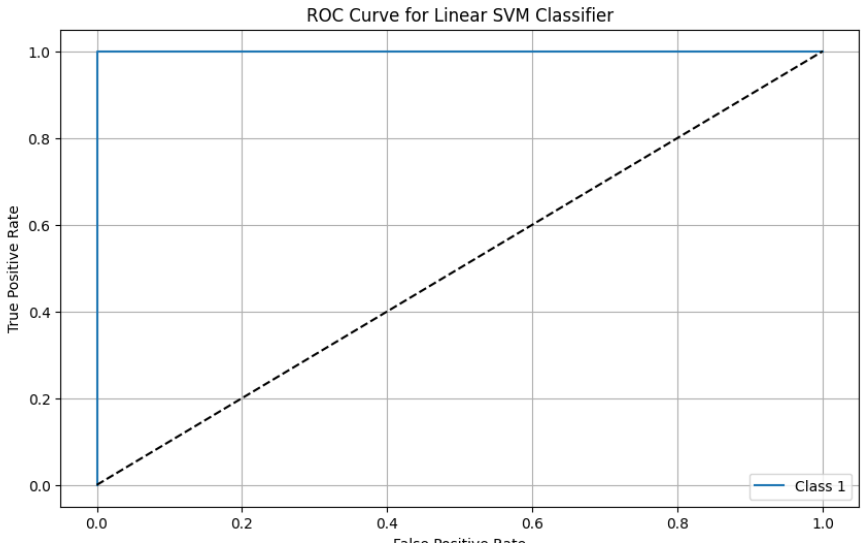
Συνεπώς, προτιμώνται **μεσαίες τιμές για το γ (0.01–0.1) και σχετικά υψηλές τιμές c (10–100).** Αυτός ο συνδυασμός εξισορροπεί την ικανότητα γενίκευσης και αποφεύγει το underfitting ή overfitting. **Άρα, θα επιλέξουμε να χρησιμοποιήσουμε για την υλοποίηση του Non-Linear SVM τη τιμή c=10 και γ=0.1**.

Αφού υλοποιήσουμε τον συγκεκριμένο Non-Linear SVM Classifier, κατασκευάζουμε τα learning curves του training και validation set. Για τη δημιουργία του ROC curve χρησιμοποιούμε την εντολή **y_prob = svm.predict_proba(X_test)[:, 1]**, η οποία εξάγει τις πιθανότητες για τη θετική κατηγορία (class 1) και στη συνέχεια πάλι θέτουμε στη συνάρτηση **roc_curve( )** τα **y_proba** και **y_test**, έτσι ώστε να επιστραφούν τα tpr και fpr. Τέλος, τα AUC και F1-Score υπολογίζονται και αυτά με το ίδιο σκεπτικό με πριν. Παρακάτω βλέπουμε τα αποτελέσματα:

Εικόνα 16: Learning curve for training and validation set



Αρχικά, το F1 Score της εκπαίδευσης είναι πολύ υψηλό και όσο το μέγεθος των δεδομένων αυξάνεται, το F1 Score της εκπαίδευσης πέφτει ελαφρώς αλλά παραμένει πολύ κοντά στο 1.0, κάτι που δείχνει ότι το μοντέλο είναι αρκετά ισχυρό και συνεχίζει να μαθαίνει σωστά από τα δεδομένα. Από την άλλη, με **μικρό μέγεθος** δεδομένων εκπαίδευσης, το F1 Score επικύρωσης είναι αρκετά χαμηλότερο υποδεικνύοντας ότι **το μοντέλο δεν γενικεύει καλά στην αρχή**. Όσο αυξάνεται το μέγεθος των δεδομένων εκπαίδευσης, το F1 Score επικύρωσης ανεβαίνει σταθερά και πλησιάζει το F1 Score εκπαίδευσης. **Στο τέλος, οι δύο γραμμές συγκλίνουν και παραμένουν κοντά στο 1.0, κάτι που δείχνει καλή γενίκευση του μοντέλου.**



Εικόνα 17: ROC curve for class 1

Η καμπύλη ROC είναι τέλεια και πλησιάζει το πάνω αριστερό μέρος του διαγράμματος, το οποίο σημαίνει ότι **ο ταξινομητής έχει εξαιρετική απόδοση**. Άρα, ο ταξινομητής **διαχωρίζει τις δύο κατηγορίες σχεδόν τέλεια** και η πιθανότητα να κάνει λάθος (False Positive ή False Negative) είναι εξαιρετικά χαμηλή, αφού δεν μπερδεύει τα αρνητικά δείγματα ως θετικά.

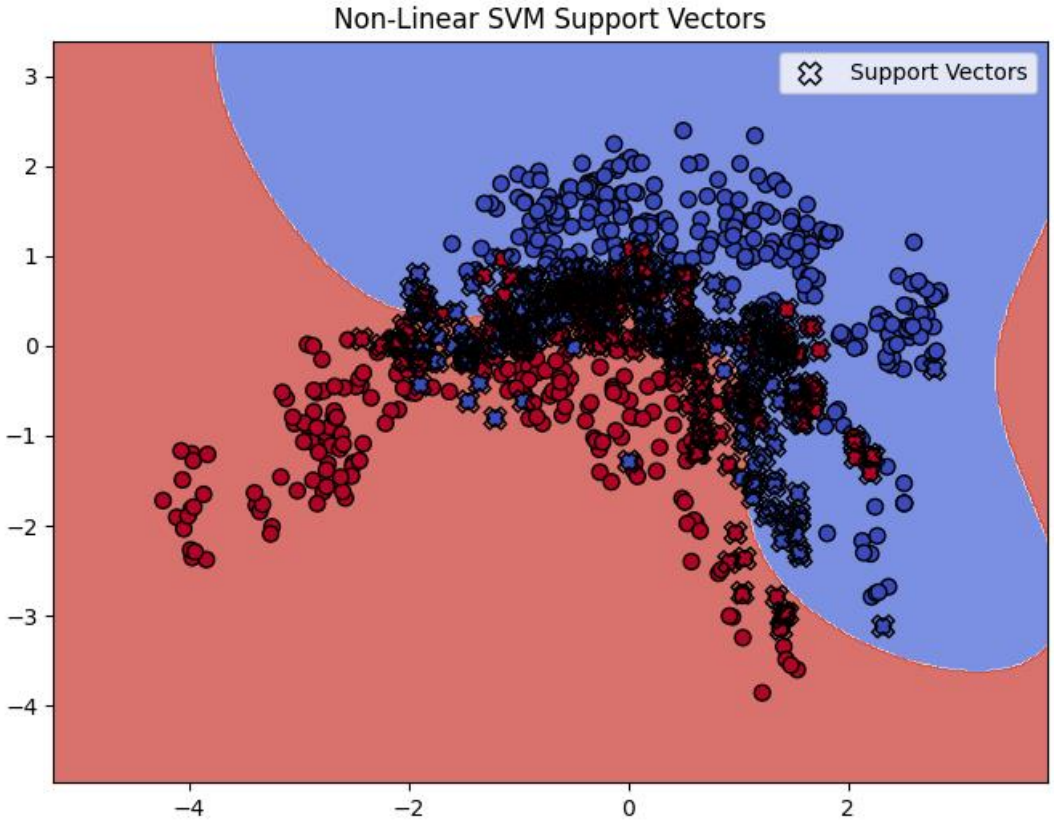
Average AUC for Linear SVM: 1.0000

Το AUC δείχνει ότι ο ταξινομητής έχει **τελεία συνολική ικανότητα διάκρισης** μεταξύ της θετικής (κλάση 1) και της αρνητικής κλάσης (κλάση 0).

F1 Score for Non-Linear SVM: 1.0000

Η τιμή του F1-Score είναι μια ακόμα ένδειξη ότι ο ταξινομητής έχει **τέλεια απόδοση**, καθώς επιτυγχάνει απόλυτη ακρίβεια και ανάκληση.

Πάλι με τη συνάρτηση **plot_support_vectors( )** επιχειρούμε να οπτικοποιήσουμε τα διανύσματα υποστήριξης στο επίπεδο. Αυτή τη φορά, **η επιφάνεια απόφασης** που εμφανίζεται (διαχωρισμός μεταξύ κόκκινων και μπλε περιοχών) **είναι μη γραμμική**, κάτι που αναμενόταν για ένα Non-Linear SVM. Το σχήμα της επιφάνειας απόφασης δείχνει ότι το μοντέλο χρησιμοποιεί τον πυρήνα (kernel) για να επιτύχει τον μη γραμμικό διαχωρισμό των κατηγοριών. Τα support vectors σε περιοχές που εμφανίζουν αλληλοεπικάλυψη των δύο κατηγοριών (όπου κόκκινοι και μπλε κύκλοι είναι κοντά) δείχνουν ότι το μοντέλο έχει εντοπίσει και προσπαθεί να διαχειριστεί τη δυσκολία στο διαχωρισμό. Ωστόσο, **η περιοχή απόφασης φαίνεται να ακολουθεί καλά τα δεδομένα, γεγονός που μας επιβεβαιώνει ότι έχουμε αποφύγει το overfitting παρά την ύπαρξη πολλών support vectors.**



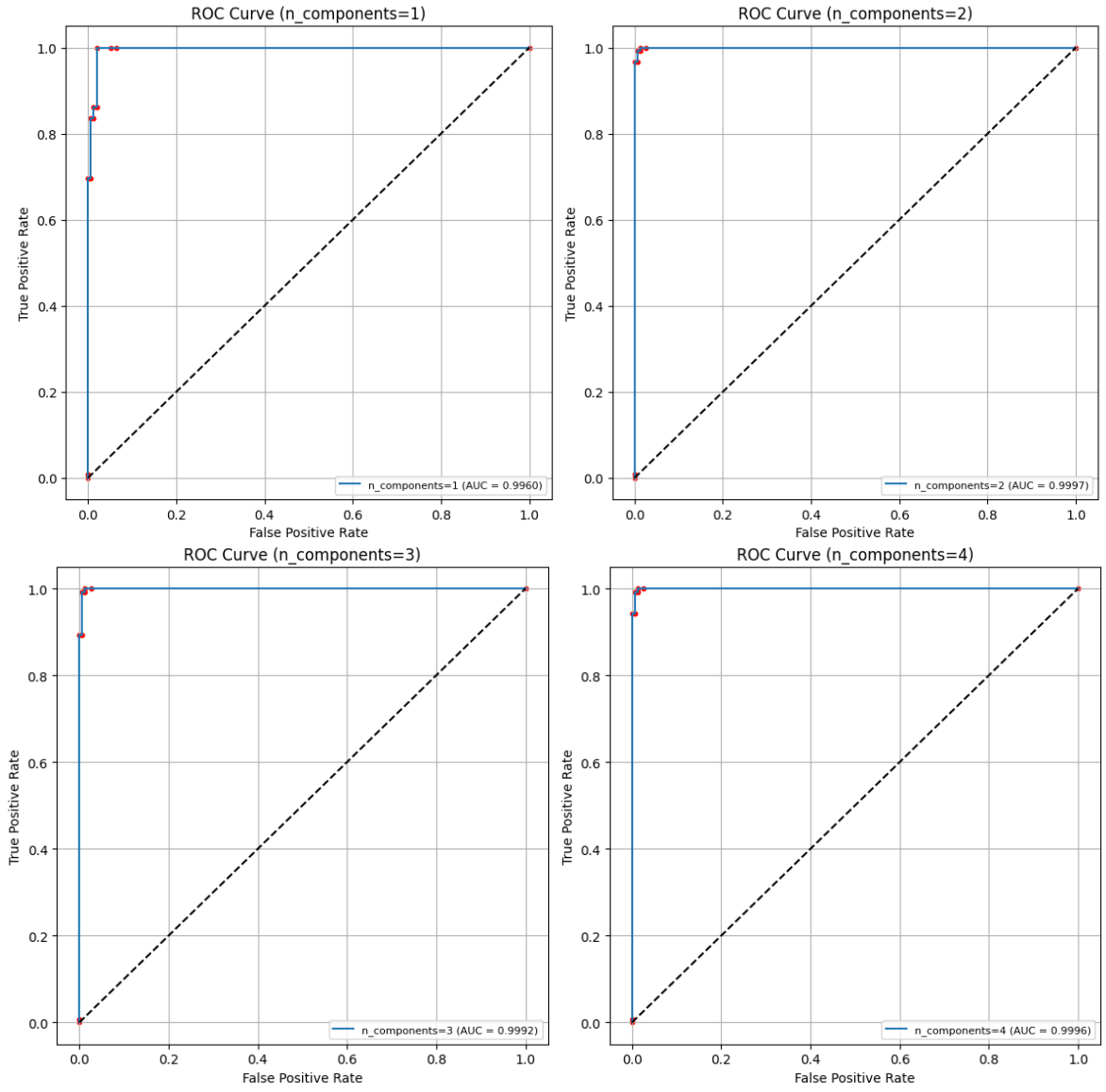
Εικόνα 18: Support vectors for Non-Linear SVM

5 Ταξινόμηση με Gaussian Mixture

Ο κατηγοριοποιητής Gaussian Mixture θεωρεί ότι τα δεδομένα προέρχονται από μία συνδυαστική κατανομή που αποτελείται από πολλές **κανονικές κατανομές (Gaussian distributions)**. Κάθε συνιστώσα (component) του GMM στην ουσία αντιστοιχεί σε έναν "υποτύπο" των θετικών δειγμάτων και τις χρησιμοποιούμε για να εκπαιδεύσουμε το μοντέλο να μάθει να περιγράφει τη δομή των θετικών δεδομένων, όπου δηλαδή το label είναι 1. Έπειτα, για κάθε δείγμα από το σύνολο ελέγχου **X_test**, το GMM υπολογίζει την πιθανότητα log-likelihood να ανήκει στα θετικά δείγματα, προκειμένου να κάνει τη κατηγοριοποίηση. Πιο συγκεκριμένα, χρησιμοποιείται το **threshold** για να καθορίσει το όριο απόφασης για το πότε θα ταξινομηθεί η παρατήρηση ως θετική, βασισμένο στις πιθανότητες που προκύπτουν από το μοντέλο.

Όσον αφορά το κώδικα, χωρίζουμε τα αρχικά κανονικοποιημένα δεδομένα χωρίς PCA πάλι σε αναλογία **60-20-20**. Έπειτα, στη μεταβλητή **X_train_positive** αποθηκεύουμε τα δεδομένα του **X_train**, τα οποία ανήκουν στη θετική κλάση (Class 1). Ορίζουμε τη συνάρτηση **gm( )**, η οποία χρησιμοποιεί το αντικείμενο **GaussianMixture( )** της βιβλιοθήκης sklearn και το κάνει **fit()** στο **X_train_positive** για τις διάφορες τιμές του **n_components**. Έπειτα, μέσω της **score_samples( )** υπολογίζουμε το log-likelihood και το αποθηκεύουμε στη μεταβλητή **prob**. Τέλος, υπολογίζουμε τα ROC curves για κάθε threshold που προκύπτει κατά τα γνωστά και χρησιμοποιούμε πλήθος συνιστωσών στο διάστημα 1-5. Παρατηρούμε το εξής:

Εικόνα 19: ROC curves for different n_components and threshold values



Όλες οι καμπύλες ROC είναι πολύ κοντά στην άνω αριστερή γωνία, που σημαίνει ότι **το μοντέλο έχει πολύ καλή απόδοση συνολικά**, ανεξαρτήτως αριθμού Gaussian συνιστωσών. Αυτό υποδεικνύει ότι το μοντέλο μπορεί να διαχωρίσει καλά τις κατηγορίες (θετικές και αρνητικές), κάτι που φαίνεται και από τη μικρή διασπορά στις καμπύλες. Άρα, **η αύξηση του αριθμού συνιστωσών (components) δεν φαίνεται να προσφέρει σημαντική βελτίωση**, καθώς το μοντέλο ήδη αποδίδει εξαιρετικά.

Από τα **thresholds** φαίνεται επίσης ότι το μοντέλο είναι **σταθερό και ευαίσθητο σε διαφορετικές τιμές**, καθώς η απόδοση (**AUC**) παραμένει σχεδόν **αμετάβλητη**. Στα πολύ χαμηλά thresholds, το μοντέλο τείνει να χαρακτηρίζει περισσότερα δείγματα ως θετικά (υψηλή TPR). Αυτό μπορεί να οδηγήσει σε λίγα ψευδώς θετικά (False Positives), αλλά το FPR παραμένει πολύ μικρό. Αυτή η συμπεριφορά είναι ευνοϊκή, καθώς σημαίνει ότι το μοντέλο μπορεί να ανιχνεύσει σχεδόν όλα τα θετικά δείγματα. Στα υψηλά thresholds από την άλλη, το μοντέλο γίνεται πιο αυστηρό, δηλαδή ταξινομεί ως θετικά μόνο τα πιο σίγουρα δείγματα. Αυτό μειώνει το TPR (λιγότερα True Positives), αλλά το FPR παραμένει εξαιρετικά χαμηλό.

6 Συμπεράσματα

Συγκρίνοντας όλα τα μοντέλα που μελετήσαμε, καταλήγουμε στο συμπέρασμα ότι **τα δύο εκείνα που επιτυγχάνουν τέλεια απόδοση** όσον αφορά AUC και F1-Score είναι ο **k-NN classifier** και ο **Non-Linear SVM classifier**. Και τα δύο μοντέλα μπόρεσαν να προσαρμοστούν πάρα πολύ καλά στις περιοχές όπου οι κλάσεις δεν ήταν γραμμικώς διαχωρίσιμες. Αυτό οφείλεται στο ότι το k-NN είναι εξαιρετικά ευαίσθητο στις τοπικές διακυμάνσεις των δεδομένων, ενώ το Non-Linear SVM χρησιμοποιεί πυρήνα για να επιτύχει μη γραμμικό διαχωρισμό, προσφέροντας ευελιξία. Επιπλέον, το **Linear SVM είναι επίσης εξαιρετικό**, αλλά παρατηρούμε ότι περιορίζεται ελάχιστα η απόδοσή του στα μη-γραμμικά διαχωρίσιμα δεδομένα, τα οποία έχουν μεγαλύτερη πολυπλοκότητα. Στη **τελευταία θέση** βρίσκεται το **Parzen Window** με χαμηλότερο F1 Score και AUC από όλα τα μοντέλα. Αυτό είναι αναμενόμενο, καθώς σε πιο σύνθετες κατανομές το Parzen Window δυσκολεύεται να διαχωρίσει τις κλάσεις, ειδικά όταν υπάρχει μεγάλη αλληλοεπικάλυψη όπως στη δικιά μας περίπτωση. Επιπλέον, το γεγονός ότι έχουμε χρησιμοποιήσει για το Parzen Window τα χαρακτηριστικά που προέκυψαν μετά την εφαρμογή του PCA δικαιολογεί ακόμα περισσότερο την χαμηλότερη απόδοση. Εφόσον το dataset μας περιέχει εξ αρχής 4 features τα οποία είναι ούτως ή άλλως λίγα σε αριθμό και μπορούν να περιγράψουν καλά τις συσχετίσεις των δεδομένων, η μείωση της διάστασης μπορεί να οδηγήσει σε απώλεια σημαντικής πληροφορίας. Στην περίπτωση αυτή, το PCA μπορεί να απλοποιήσει τα δεδομένα μας με τέτοιο τρόπο που να μην διατηρεί τα κρίσιμα χαρακτηριστικά.

Παρακάτω παρατίθενται συγκεντρωμένες όλες οι μετρικές που αξιοποιήσαμε για την αξιολόγηση των μοντέλων:

	Parzen Window	**k-NN**	**Linear SVM**	**Non-Linear SVM**
F1-Score	0.8151	1.000	0.9964	1.000
AUC	0.7746	1.000	0.9999	1.000

7 Παράρτημα

Παρατίθεται το Github Repository με το συνολικό κώδικα, σχόλια και επεξηγήσεις στο αντίστοιχο Jupyter Notebook:
https://github.com/miltiadiss/CEID_NE5237-Decision-Theory/tree/main/Project%202

Επίσης, παρατίθενται οι πηγές και η βιβλιογραφία που χρησιμοποιήθηκαν κατά τη μελέτη για την υλοποίηση της εργασίας:
https://eclass.upatras.gr/modules/document/?course=CEID1039
https://www.javatpoint.com/parzen-windows-density-estimation-technique
https://scikit-learn.org/1.5/modules/svm.html
https://scikit-learn.org/1.5/modules/mixture.html
https://www.w3schools.com/python/python_ml_knn.asp