2Η ΕΡΓΑΣΙΑ ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ

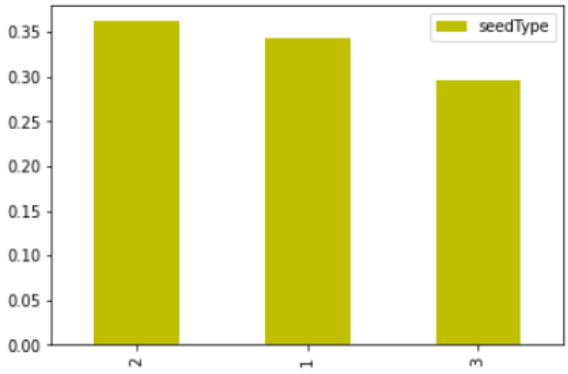
Κατηγοριοποίηση Συνόλου Δεδομένων

Σταυρούλα Κρανίτη(Ε19074)
Τμήμα Ψηφιακών Συστημάτων
Πανεπιστήμιο Πειραιώς
stavkraniti01@gmail.com

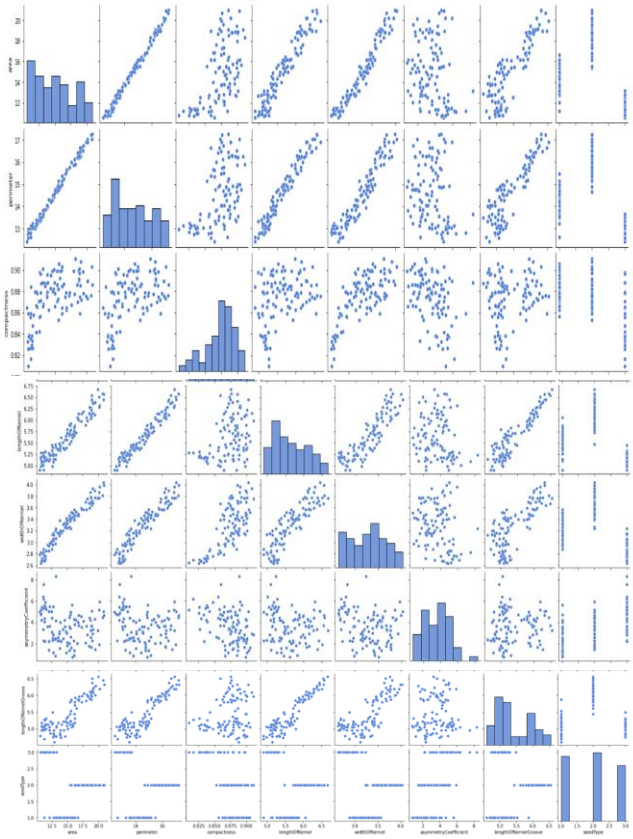
Το σύνολο δεδομένων που αντιστοιχεί στο τελευταίο ψηφίο του αριθμού μητρώου μου είναι αυτό με το νούμερο 4. Τα δεδομένα αφορούν μετρήσεις γεωμετρικών ιδιοτήτων πυρήνων που ανήκουν σε τρεις διαφορετικές ποικιλίες σίτου. Η εξεταζόμενη ομάδα περιελάβανε πυρήνες που ανήκουν σε τρεις διαφορετικές ποικιλίες σιταριού: Kama, Rosa και Canadian, 70 στοιχεία το καθένα, επιλεγμένα τυχαία για το πείραμα. Ανιχνεύθηκε υψηλής ποιότητας οπτικοποίηση της εσωτερικής δομής του πυρήνα χρησιμοποιώντας μια τεχνική μαλακής ακτινογραφίας. Στην εργασία μας ζητείται να γίνει κατηγοριοποίηση (classification) του συνόλου δεδομένων.
Το πρώτο βήμα που πρέπει να γίνει είναι η επεξεργασία του αρχείου δεδομένων που έχουμε. Πρώτα από όλα εισάγουμε πάνω από κάθε στήλη στο αρχείο μας την αντίστοιχη ετικέτα που της αντιστοιχεί με βάσει τα στοιχεία που μας δίνει η σελίδα από την οποία κατεβάσαμε τα δεδομένα. Ύστερα μετατρέπουμε το αρχείο txt αυτό σε αρχείο csv και ύστερα προχωράμε στο προγραμματιστικό κομμάτι.

Αρχικά εισάγουμε τις απαραίτητες βιβλιοθήκες για τους αλγορίθμους τις κατηγοριοποίησης. Στην συνέχεια, βρίσκουμε την ποσότητα των σπόρων που αναλογούν σε κάθε είδος καρπού από τα 3. Αφού γίνει αυτό, ακολουθούν δύο εξίσου σημαντικά βήματα που μας βοηθούν πολύ στην κατανόηση των δεδομένων μας. Χρησιμοποιούνται οι συναρτήσεις info() και describe(). Η info μας δίνει των τύπο των δεδομένων που υπάρχουν σε κάθε μία από τις στήλες των δεδομένων. Η describe χρησιμοποιείται για την προβολή ορισμένων βασικών στατιστικών λεπτομερειών όπως εκατοστημόριο, μέσος όρος, std κ.λπ. ενός πλαισίου δεδομένων ή μιας σειράς αριθμητικών τιμών.

Ένα ακόμα βήμα για την περεταίρω κατανόηση και ανάλυση των δεδομένων μας είναι η αναπαράσταση. Χρησιμοποιούμε το df['seedType'] για να αποκτήσουμε πρόσβαση στο πεδίο seedType και καλούμε την value_counts για να λάβουμε ένα πλήθος μοναδικών τιμών. Αυτές τις τιμές τις κανονικοποιούμε. Η κανονικοποίηση των δεδομένων χρησιμοποιείται με σκοπό η εκπαίδευση μοντέλων να είναι λιγότερο ευαίσθητη στην κλίμακα των χαρακτηριστικών. Αυτό επιτρέπει στο μοντέλο μας να συγκλίνει καλύτερα και, με τη σειρά του, οδηγεί σε ένα πιο ακριβές μοντέλο. Ύστερα καλούμε τη μέθοδο plot και μεταβιβάζουμε στο bar (αφού θέλουμε γράφημα τύπου μπάρας), το όρισμα. Το σχήμα που προκύπτει είναι το εξής:

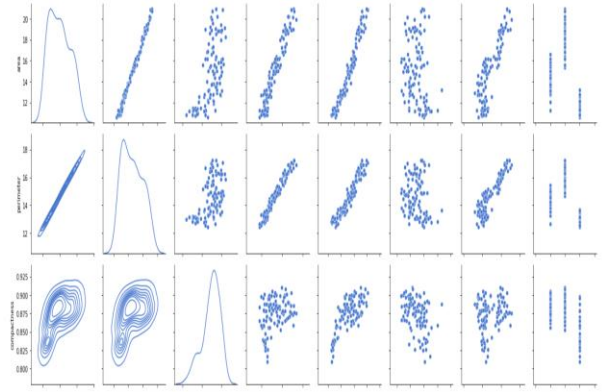


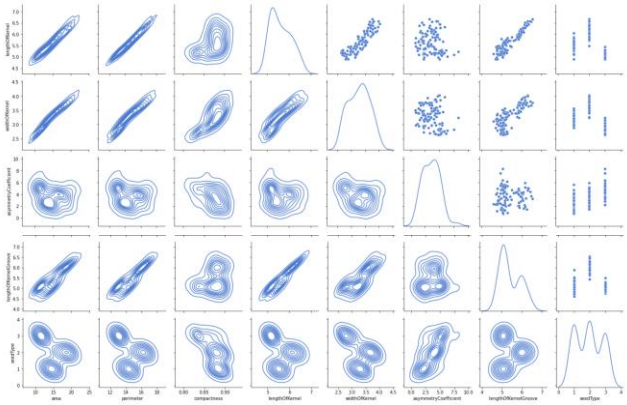
Το επόμενο βήμα θα είναι να υπολογίσουμε τον συντελεστή συσχέτισης. Ένας τρόπος για τη διερεύνηση της συσχέτισης μεταξύ δύο μεταβλητών είναι να δημιουργήσουμε ένα διάγραμμα διασποράς όπου απεικονίζουμε ζεύγη παρατηρήσεων. Αν πραγματοποιήσουμε τη συγκεκριμένη διαδικασία για όλους τους συνδυασμούς στηλών (μεταβλητών) μπορούμε να αναπαραστήσουμε γραφικά τη συσχέτιση τους. Με τον τρόπο αυτό έχουμε τη δυνατότητα να παρατηρήσουμε γραφικά τη συσχέτιση μεταξύ δύο μεταβλητών. Θα δημιουργήσουμε διαγράμματα, γι'αυτό το σκοπό, με τη βοήθεια της βιβλιοθήκης seaborn. Η seaborn είναι μία βιβλιοθήκη οπτικοποίησης δεδομένων της Python που βασίζεται στην matplotlib. Πιο συγκεκριμένα θα χρησιμοποιήσουμε το PairGrid. Αυτό το αντικείμενο χαρτογραφεί κάθε μεταβλητή σε ένα σύνολο δεδομένων σε μια στήλη και μια σειρά σε ένα πλέγμα πολλαπλών αξόνων. Καθώς όμως θέλουμε να αναπαραστήσουμε πάρα πολλά δεδομένα, θα χρησιμοποιήσουμε την μέθοδο pairplot(), όπου ακριβώς, εξυπηρετεί την αναπαράσταση πολλών γραφικών παραστάσεων σε μία γραμμή. Το σχήμα που προκύπτει είναι το παρακάτω.



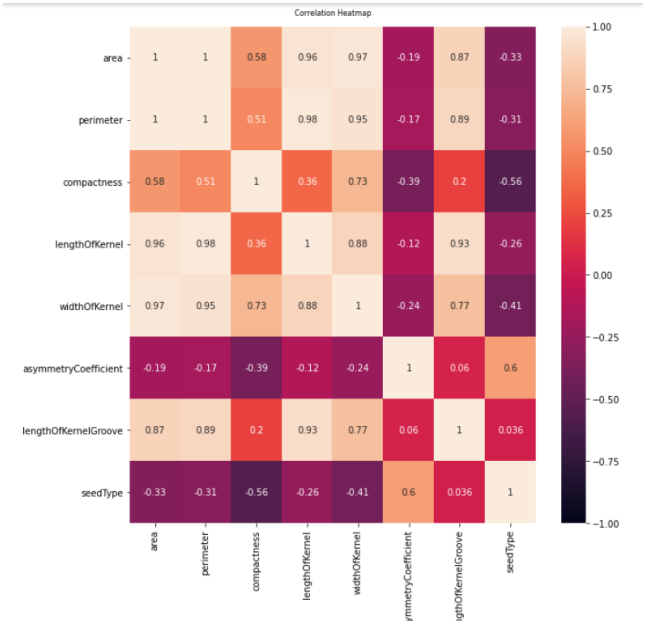
(Λόγω μεγέθους παραθέτεται σε 3 μέρη.)

Είναι επίσης δυνατό να χρησιμοποιηθούν διαφορετικές συναρτήσεις στα άνω και κάτω τρίγωνα του παραπάνω plot (οι οποίες σε κάποια άλλη περίπτωση θα ήταν περιττές). Το αντίστοιχο σχήμα με αυτές τις συναρτήσεις παραθέτεται παρακάτω.





Ένας ακόμα τρόπος για την εύρεση της συσχέτισης μεταξύ των στηλών του πλαισίου δεδομένων μπορούμε να υπολογίσουμε το συντελεστή γραμμικής συσχέτισης (συντελεστής Pearson). Ο συγκεκριμένος συντελεστής μπορεί να εκτιμήσει το βαθμό γραμμικής συσχέτισης μεταξύ των μεταβλητών και κυμαίνεται μεταξύ του -1 (πλήρης γραμμική αρνητική συσχέτιση) μέχρι το +1 (πλήρης γραμμική θετική συσχέτιση), με την τιμή μηδέν να αντιστοιχεί σε μη συσχετισμένες μεταβλητές (γραμμικά). Για τον υπολογισμό της συσχέτισης μεταξύ των διαφορετικών στηλών (μεταβλητών) χρησιμοποιείται η συνάρτηση corr() της Pandas για τις διαφορετικές παραμέτρους. Παρακάτω φαίνεται το heatmap που δείχνει καθαρά την συσχέτηση.



Η συσχέτιση κυμαίνεται από -1 έως +1. Οι τιμές που πλησιάζουν το μηδέν σημαίνει ότι δεν υπάρχει γραμμική τάση μεταξύ των δύο μεταβλητών. Όσο πιο κοντά στο 1 είναι η συσχέτιση, τόσο πιο θετική είναι. Καθώς αυξάνεται το ένα, τόσο αυξάνεται και το άλλο και όσο πιο κοντά στο 1 βρίσκεται, τόσο ισχυρότερη είναι η σχέση. Μια συσχέτιση πλησιέστερη στο -1 είναι παρόμοια, αλλά

αντί να αυξάνονται και οι δύο, η μία θα αυξάνεται ενώ η άλλη θα μειώνεται. Οι διαγώνιες είναι όλες 1. Αυτά τα τετράγωνα συσχετίζουν κάθε μεταβλητή με τον εαυτό της. Για τα υπόλοιπα, όσο μεγαλύτερος είναι ο αριθμός και πιο ανοιχτό είναι το χρώμα, τόσο υψηλότερη είναι η συσχέτιση μεταξύ των δύο μεταβλητών. Το διάγραμμα είναι, επίσης, συμμετρικό για την κύριο διάμετρο διαγώνιο, καθώς οι ίδιες δύο μεταβλητές βρίσκονται μαζί στο ίδιο κουτάκι.

Στην συνέχεια ακολουθεί η προεπεξεργασία των δεδομένων. Για την προετοιμασία των δεδομένων ένας από τους πιο χρήσιμους μετασχηματισμούς είναι η κλιμάκωση των δεδομένων. Η διαδικασία αυτή μετατρέπει τα δεδομένα με τέτοιο τρόπο ώστε οι τιμές των διαφορετικών στηλών (μεταβλητών) να μεταβάλλονται εντός συγκεκριμένου διαστήματος. Με τον τρόπο αυτό μειώνεται η επίδραση από τη διαφορά στις αριθμητικές κλίμακες των δεδομένων (π.χ. μια μεταβλητή μπορεί να μεταβάλλεται στο διάστημα 0-1 και μια άλλη στο διάστημα -1000 έως 10000000) στους αλγόριθμους μηχανικής μάθησης. Εισάγουμε από την scikit-learn την συνάρτηση StandardScaler(). Θα χρησιμοποιηθεί η μέθοδος της τυποποίησης για τα δεδομένα. Το κεντράρισμα και η κλιμάκωση πραγματοποιούνται ανεξάρτητα σε κάθε λειτουργία υπολογίζοντας τα σχετικά στατιστικά στοιχεία για τα δείγματα στο training set. Στη συνέχεια αποθηκεύονται η μέση και η τυπική απόκλιση για χρήση σε μεταγενέστερα δεδομένα χρησιμοποιώντας μετασχηματισμό.

Έπειτα συνεχίζουμε την δημιουργία συνόλων εκπαίδευσης και ελέγχου. Σκοπός είναι να εκπαιδευτεί ο αλγόριθμος ώστε να ανταποκρίνεται με σωστό τρόπο σε μελλοντικά δεδομένα (αποτελεσματικότητα στην πραγματοποίηση ορθών προβλέψεων). Θα χρησιμοποιήσουμε τη συνάρτηση train_test_split() της scikit-learn. Ουσιαστικά, χωρίζει τους πίνακες σε τυχαία train και test υποσύνολα. Το training set είναι ένα υποσύνολο για να 'εκπαιδεύει' το μοντέλο. Το test set είναι ένα υποσύνολο για να τεστάρει το εκπαιδευμένο μοντέλο.

Μετά από αυτό ακολουθούν 3 αλγόριθμοι κατηγοριοποίησης. Οι αλγόριθμοι που έχουν χρησιμοποιηθεί είναι οι εξής:

- Τυχαίο Δάσος (random forest)
- Γκαουσιανός Ταξινομητής Bayes (Gaussian Naïve Bayes)
- Πλησιέστεροι γείτονες (k-nearest neighbors)

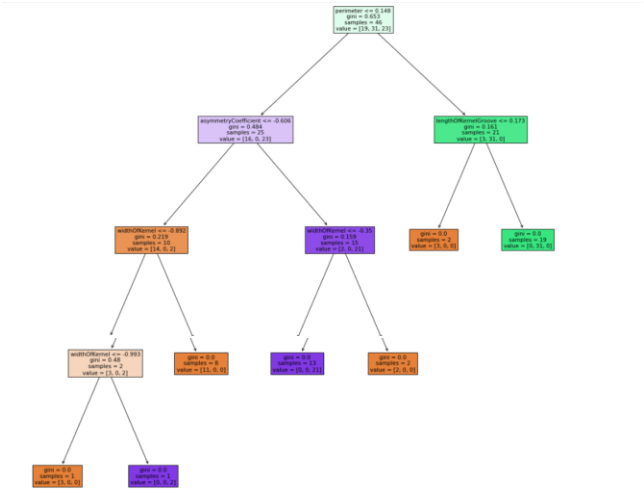
Ας αναλύσουμε αρχικά τον πρώτο αλγόριθμο, αυτό του τυχαίου δάσους. Ένα τυχαίο δάσος είναι ένας εκτιμητής που ταιριάζει σε έναν αριθμό ταξινομητών αποφάσεων δέντρων σε διάφορα υποσύνολα του συνόλου δεδομένων και χρησιμοποιεί τον μέσο όρο για τη βελτίωση της ακρίβειας της πρόγνωσης και του ελέγχου της υπερπροσαρμογής. Το τυχαίο δάσος, όπως υποδηλώνει το όνομά του, αποτελείται από μεγάλο αριθμό μεμονωμένων δέντρων απόφασης που λειτουργούν ως σύνολο . Κάθε μεμονωμένο δέντρο στο τυχαίο δάσος ξετυλίγει μια πρόβλεψη τάξης και η τάξη με τις περισσότερες ψήφους γίνεται η πρόβλεψη του μοντέλου μας. Ο συγκεκριμένος αλγόριθμος λειτουργεί πολύ καλά καθώς ένας μεγάλος αριθμός σχετικά μη συσχετισμένων μοντέλων (δέντρα) που λειτουργούν ως επιτροπή θα ξεπεράσουν οποιοδήποτε από τα μεμονωμένα συστατικά μοντέλα. μερικά δέντρα μπορεί να είναι λάθος, πολλά άλλα δέντρα θα είναι σωστά, έτσι ως ομάδα τα δέντρα μπορούν να κινηθούν προς τη σωστή κατεύθυνση. κάθε δέντρο σε ένα τυχαίο δάσος μπορεί να επιλέξει μόνο από ένα τυχαίο υποσύνολο χαρακτηριστικών.

Αυτό επιβάλλει ακόμη μεγαλύτερη διακύμανση μεταξύ των δέντρων στο μοντέλο και τελικά οδηγεί σε χαμηλότερη συσχέτιση μεταξύ των δέντρων και περισσότερη διαφοροποίηση. Στον κώδικα συμβαίνει το εξής. Αρχικά, εισάγουμε από την βιβλιοθήκη sklearn.model_selection την συνάρτηση StandardScaler(). Η συνάρτηση StandardScaler θα μεταμορφώσει τα δεδομένα έτσι ώστε η κατανομή να έχει μέση τιμή 0 και τυπική απόκλιση 1. Ύστερα προσαρμόζουμε τα δεδομένα ώστε μετά να τα μετατρέψουμε. Εκπαιδεύουμε τα δεδομένα και τα τεστάρουμε και μετά συνεχίζουμε με το τυχαίο δάσος. Εισάγουμε τις κατάλληλες βιβλιοθήκες και χρησιμοποιούμε την συνάρτηση RandomForestClassifier με σκοπό να καθοριστούν οι εκτιμητές. Φτιάχνουμε το δάσος με τα δένδρα από το σύνολο με τις εκπαιδευμένες μεταβλητές και έπειτα με τις συναρτήσεις predict() και predict_proba() υπολογίζουμε την τάξη και τις πιθανότητες τάξεις των εκπαιδευμένων x. Η ίδια διαδικασία επαναλαμβάνεται και για τα test_X. Εκτυπώνουμε και για τις δυο τον πίνακα σύγχυσης καθώς και τη βαθμολογία ακρίβειας. Ο πίνακας σύγχυσης αξιολογεί τη ποιότητα της εξόδου ενός ταξινομητή. Τα διαγώνια στοιχεία αντιπροσωπεύουν τον αριθμό των σημείων για τα οποία η προβλεπόμενη ετικέτα είναι ίση με την πραγματική ετικέτα, ενώ τα υπόλοιπα στοιχεία εκτός της κύριας διαγωνίου είναι αυτά που δεν έχουν επισημανθεί σωστά από τον ταξινομητή. Όσο υψηλότερες είναι οι διαγώνιες τιμές του πίνακα σύγχυσης τόσο πιο πολλές σωστές προβλέψεις έχουμε. Από την άλλη ο υπολογισμός ακριβείας υπολογίζεται ως εξής. Δεδομένων δύο λιστών, y_pred και y_true, για κάθε δείκτη θέσης i, συγκρίνετε το i στοιχείο του y_pred με το i στοιχείο του y_true και αρχικά μετράει τον αριθμών των κοινών στοιχείων και διαιρέστε το με τον αριθμό των δειγμάτων. Για το σύνολο δεδομένων που έχουμε εμείς το αποτέλεσμα που εμφανίζεται είναι το παρακάτω.

```
Confusion Matrix - Train:
 [[22  0  0]
 [ 0 30  0]
 [ 0  0 21]]


Overall Accuracy - Train:  1.0


Confusion Matrix - Test:
 [[11  1  2]
 [ 0  8  0]
 [ 2  0  8]]


Overall Accuracy - Test:  0.84375
```

Όπως φαίνεται στην εικόνα με βάση το πίνακα σύγχυσης τόσο για τα train όσο και για τα test στοιχεία η διαγώνιος αποτελείται από αρκετά μεγάλους αριθμούς, το οποίο υποδηλώνει πως έχουμε πολλές σωστές προβλέψεις.

Με βάση τη βαθμολογία εγκυρότητας για τα train στοιχεία έχουμε εγκυρότητα 1.0 (δηλαδή 100%), το οποίο σημαίνει πως έχουμε έγκυρα αποτελέσματα.
Για τα στοιχεία test έχουμε βαθμό εγκυρότητας 0.84 (δηλαδή 84%), το οποίο είναι ένα εξίσου μεγάλο ποσοστό εγκυρότητας. Έπειτα εκτυπώνουμε το Decision Tree, το οποίο παραθέτεται παρακάτω.



Στην συνέχεια ακολουθεί ο αλγόριθμος Gaussian Naive Bayes για κατηγοριοποίηση. οι αφελείς ταξινομητές Bayes είναι μια οικογένεια απλών "πιθανολογικών ταξινομητών" που βασίζονται στην εφαρμογή του θεωρήματος του Bayes με ισχυρές (αφελείς) παραδοχές ανεξαρτησίας μεταξύ των χαρακτηριστικών. Γενικότερα οι μέθοδοι Naive Bayes είναι ένα σύνολο εποπτευόμενων αλγορίθμων μάθησης που βασίζονται στην εφαρμογή του θεωρήματος του Bayes με την «αφελής» υπόθεση της υπό όρους ανεξαρτησίας μεταξύ κάθε ζεύγους χαρακτηριστικών δεδομένης της τιμής της μεταβλητής κλάσης. Οι αφελείς ταξινομητές Bayes λειτούργούν αρκετά καλά σε πολλές πραγματικές καταστάσεις, όπως η ταξινόμηση εγγράφων και το φιλτράρισμα ανεπιθύμητων μηνυμάτων (spam). Απαιτούν μια μικρή ποσότητα δεδομένων εκπαίδευσης για την εκτίμηση των απαραίτητων παραμέτρων. Οι learners και οι classifiers του Naive Bayes είναι αρκετά γρήγοροι σε σύγκριση με άλλες μεθόδους. Κάθε κατανομή μπορεί να εκτιμηθεί ανεξάρτητα ως μονοδιάστατη κατανομή. Αυτό βοηθά στην επίλυση των προβλημάτων που απορρέουν λόγω των διαστημάτων. Παρόλα ταύτα οι αφελείς Bayes μπορεί να είναι καλοί classifiers όμως σαν εκτιμητές δεν είναι εξίσου καλοί. Στον κώδικα αρχίζουμε για ακόμη μία φορά με την εκπαίδευση και τις δοκιμές των δεδομένων αφού έχουμε εισάγει πρώτα από την βιβλιοθήκη sklearn.naive_bayes την εξίσωση GaussianNB (η οποία χρησιμοποιείται για το classification στην Naive Bayes. Ύστερα βρίσκουμε με τις συναρτήσεις predict() και predict_proba() τις προβλέψεις για τα training και testing sets και όπως και παραπάνω τα εμφανίζουμε.
Ακολουθεί εικόνα των αποτελεσμάτων.

```
Confusion Matrix - Train:
 [[21  0  1]
 [ 1 29  0]
 [ 2  0 19]]


Overall Accuracy - Train:  0.9452054794520548


Confusion Matrix - Test:
 [[12  1  1]
 [ 0  8  0]
 [ 0  0 10]]


Overall Accuracy - Test:  0.9375
```

Όπως φαίνεται και από τον πίνακα σύγχυσης και από την βαθμολογία εγκυρότητας, έχουμε πάρα πολλές καλές προβλέψεις. Ωστόσο ο συγκεκριμένος αλγόριθμος δεν είναι ιδιαίτερα αξιόπιστος όσο έχει κάνει με προβλέψεις. Βέβαια, με βάση τον προηγούμενο αλγόριθμο, όντως έχουμε αρκετές σωστές προβλέψεις. Ο λόγος χρήσης αυτού του αλγορίθμου είναι ουσιαστικά για να ελεγχθεί κατά πόσο είναι μη αξιόπιστος, συγκρίνοντας από αποτελέσματα των δύο αλγορίθμων.

Τέλος έχουμε τον αλγόριθμο του πλησιέστερου γείτονα. Η κατηγοριοποίηση του πλησιέστερου γείτονα είναι τμήμα μιας πιο γενικής τεχνικής, που είναι γνωστή ως εκπαίδευση βάσει στιγμιότυπων, η οποία δεν κατασκευάζει ένα γενικό μοντέλο, αλλά χρησιμοποιεί συγκεκριμένα δείγματα εκπαίδευσης για να κάνει προβλέψεις για ένα στιγμιότυπο ελέγχου. Τέτοιοι αλγόριθμοι απαιτούν ένα μέτρο εγγύτητας για να προσδιορίζεται η ομοιότητα ή η απόσταση μεταξύ των στιγμιοτύπων και μια συνάρτηση κατηγοριοποίησης, που επιστρέφει την προβλεπόμενη κατηγορία ενός στιγμιοτύπου ελέγχου με βάση την εγγύτητα του με άλλα στιγμιότυπα.

Οι κατηγοριοποιητές πλησιέστερου γείτονα μπορούν να παράγουν εσφαλμένες προβλέψεις, εκτός αν χρησιμοποιηθεί το κατάλληλο μέτρο εγγύτητας και γίνουν βήματα προεπεξεργασίας των δεδομένων.

Στον κώδικα, αρχικά εισάγουμε τις κατάλληλες βιβλιοθήκες και ύστερα βρίσκουμε τους κοντινότερους γείτονες ενός σημείου. Υποδεικνύουμε τις παραμέτρους και δημιουργούμε μια μεταβλητή για την εύρεση των πλησιέστερων γειτόνων και στην συνέχεια τους εκτυπώνουμε. Με την συνάρτηση fit() προσαρμόζουμε τον κατηγοριοποιητή των πλησιέστερων γειτόνων στα στοιχεία που έχουμε εκπαιδεύσει. Μετά από αυτό

κάνουμε χρήση για άλλη μία φορά των συναρτήσεων predict() και predict_proba() ώστε να υπολογίζουμε την τάξη και τις πιθανότητες τάξεις των εκπαιδευμένων x. Η ίδια διαδικασία επαναλαμβάνεται και για τα test_X. Εκτυπώνουμε και για τις δυο τον πίνακα σύγχυσης καθώς και τη βαθμολογία ακρίβειας.

```
Confusion Matrix - Train:
 [[21  1  0]
 [ 1 29  0]
 [ 4  0 17]]


Overall Accuracy - Train:  0.9178082191780822


Confusion Matrix - Test:
 [[13  1  0]
 [ 0  8  0]
 [ 1  0  9]]


Overall Accuracy - Test:  0.9375
```

Καθώς έχει γίνει η κατάλληλη προεπεξεργασία και εκπαίδευση των δεδομένων ο αλγόριθμος είναι αρκετά έμπιστος. Όπως φαίνεται στην εικόνα με βάση το πίνακα σύγχυσης τόσο για τα train όσο και για τα test στοιχεία η διαγώνιος αποτελείται από αρκετά μεγάλους αριθμούς, το οποίο υποδηλώνει πως έχουμε πολλές σωστές προβλέψεις.

Με βάση τη βαθμολογία εγκυρότητας για τα train στοιχεία έχουμε εγκυρότητα 0.91 (δηλαδή 91%), το οποίο σημαίνει πως έχουμε έγκυρα αποτελέσματα. Για τα στοιχεία test έχουμε βαθμό εγκυρότητας 0.93 (δηλαδή 93%), το οποίο είναι ένα εξίσου μεγάλο ποσοστό εγκυρότητας.
Έπειτα από αυτό, εκτυπώνουμε με την χρήση της συνάρτησης classification_report(), τις κύριες μετρήσεις κατηγοριοποίησης για αυτόν τον αλγόριθμο.

```
Classification Report-Test:
              precision    recall  f1-score

           1       0.93      0.93      0.93
           2       0.89      1.00      0.94
           3       1.00      0.90      0.95

    accuracy                           0.94
   macro avg       0.94      0.94      0.94
weighted avg       0.94      0.94      0.94
```

Precision (ακρίβεια) είναι η ικανότητα του κατηγοριοποιητή να μην επισημαίνει ως θετικό, ένα δείγμα που είναι αρνητικό.

Recall (ανάκληση) είναι η ικανότητα του κατηγοριοποιητή να βρει όλα τα θετικά δείγματα.

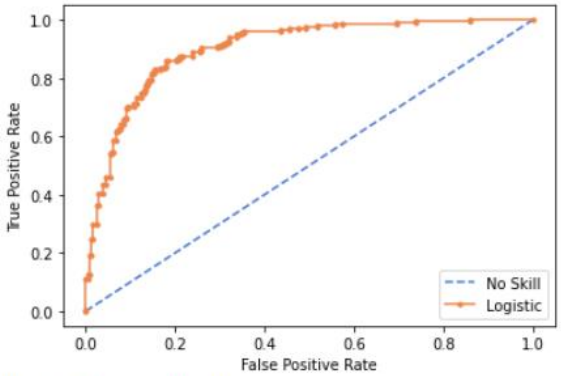
F1-score είναι ένας μέσος όρος της ακρίβειας και της ανάκλησης (precision & recall). Η καλύτερη τιμή φτάνει στο 1 και χειρότερο στο 0.
Support είναι ο αριθμός εμφανίσεων κάθε κλάσης y_true.

Accuracy υπολογίζει την ακρίβεια υποσυνόλου: το σύνολο των ετικετών που προβλέπεται για ένα δείγμα πρέπει να ταιριάζει ακριβώς με το αντίστοιχο σύνολο ετικετών στο y_true.

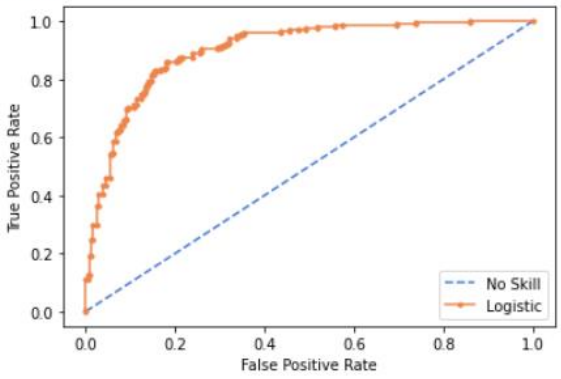
Macro avg υπολογίζει τις μετρήσεις για κάθε ετικέτα και βρίσκει τον μη σταθμισμένο μέσο όρο.

Weighted avg υπολογίζει τις μετρήσεις για κάθε ετικέτα και βρίσκει τη μέση στάθμιση τους βάσει του support.

Στην συνέχεια εκτυπώνω την roc γραφική παράσταση η οποία είναι η εξής:




```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.903
```



Οι καμπύλες ROC διαθέτουν συνήθως πραγματικό θετικό ρυθμό στον άξονα Y και ψευδώς θετικό ρυθμό στον άξονα X. Αυτό σημαίνει ότι η επάνω αριστερή γωνία της γραφικής είναι το «ιδανικό» σημείο – ένα ψευδώς θετικό ποσοστό μηδέν, και ένα πραγματικό θετικό ποσοστό ένα. Αυτό δεν είναι πολύ ρεαλιστικό, αλλά αυτό σημαίνει ότι μια μεγαλύτερη περιοχή κάτω από την καμπύλη (AUC) είναι συνήθως καλύτερη. Η «απόκλιση» των καμπυλών ROC είναι επίσης σημαντική, δεδομένου ότι είναι ιδανικό να μεγιστοποιήσετε το πραγματικό θετικό ποσοστό ενώ παράλληλα ελαχιστοποιείται το ψευδώς θετικό ποσοστό.

REFERENCES

[1] https://scikit-learn.org/stable/index.html
[2] https://developers.google.com/machine-learning/crash-course/classification/accuracy
[3] https://en.wikipedia.org/
[4] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[5] https://mljar.com/blog/visualize-tree-from-random-forest/
[6] https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
[7] https://www.educative.io/edpresso/data-normalization-in-python
[8] https://stats.stackexchange.com/questions/392517/how-can-one-interpret-a-heat-map-plot
[9] https://www.kaggle.com/abinayaravi/seeds-data-clustering-and-classification
[10] https://seaborn.pydata.org/generated/seaborn.PairGrid.html
[11] https://seaborn.pydata.org/tutorial/color_palettes.html
[12] https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html
[13] https://www.pluralsight.com/guides/data-visualization-deep-learning-model-using-matplotlib
[14] https://www.shanelynn.ie/using-pandas-dataframe-creating-editing-viewing-data-in-python/#preview-dataframes-with-head-and-tail
[15] https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html
[16] https://www.codecademy.com/articles/seaborn-design-ii
[17] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, 2018. Εισαγωγή στην Εξόρυξη Δεδομένων-2η Έκδοση