Εργαστηριακή Άσκηση για το μάθημα της Θεωρίας Αποφάσεων

Χειμερινό Εξάμηνο 2023-24

Θέμα:"Σχεδιασμός Ταξινομητή για Κατηγοριοποίηση και Πρόβλεψη Καρκίνου του Μαστού"

ΑΜΙΤΣΗΣ ΛΕΥΤΕΡΗΣ 1072464

Σε αυτή την εργαστηριακή άσκηση καλείστε να χρησιμοποιήσετε ένα σύνολο δεδομένων για να εκπαιδεύσετε ένα σύνολο ταξινομητών για την κατηγοριοποίηση και πρόβλεψη του καρκίνου του μαστού.

Το σύνολο δεδομένων που σας δίνετε στο αρχείο BreastTissue.xls παρέχει ένα σύνολο βιοιατρικών μετρήσεων για κάθε ασθενή καθώς και την πληροφορία για τον αν ο ιστός του μαστού είναι Καρκίνωμα (Carcinoma), Ινο-αδένωμα (Fibro-adenoma), Μαστοπάθεια (Mastopathy), Αδενικός (Glandular), Συνδετικός (Connective), Λιπώδης (Adipose). Κάθε γραμμή στο αρχείο αυτό περιέχει πληροφορία για διαφορετικό ασθενή. Η πρώτη στήλη αφορά τον κωδικό της κάθε καταγραφής ενώ η δεύτερη (status) δείχνει τη διάγνωση, δηλ. αν η καταγραφή αφορά ασθενή με: Καρκίνωμα (Carcinoma), Ινο-αδένωμα (Fibro-adenoma), Μαστοπάθεια (Mastopathy), Αδενικός (Glandular), Συνδετικός (Connective), Λιπώδης (Adipose).

Car	Carcinoma
Fad	Fibro-adenoma
Mas	Mastopathy
Gla	Glandular
Con	Connective
Adi	Adipose

Όλες οι άλλες στήλες αντιστοιχούν σε βιοιατρικές μετρήσεις που θα πρέπει να χρησιμοποιήσετε σαν εισόδους στους ταξινομητές που θα δημιουργήσετε. Για περισσότερες πληροφορίες για τα δεδομένα αυτά μπορείτε να δείτε την αναλυτική περιγραφή τους στην βάση δεδομένων μηχανικής μάθησης UCI (https://archive.ics.uci.edu/dataset/192/breast+tissue) από την οποία προέρχονται. Σε αυτό το link, θα βρείτε μια σύντομη περιγραφή του αρχείου (χαρακτηριστικά, πλήθος, κλπ.)

Ερώτημα 1. Προεπεξεργασία δεδομένων (10 μόρια)

Το εύρος τιμών των δεδομένων που σας έχουν δοθεί διαφέρει σημαντικά ανά χαρακτηριστικό. Για αυτό τον λόγο, για να μην υπερεκτιμηθεί η συνεισφορά κάποιου χαρακτηριστικού έναντι άλλων, θα πρέπει πριν την επεξεργασία των χαρακτηριστικών εισόδου να κανονικοποιηθούν στο εύρος [-1,1]. Χρησιμοποιήστε το matlab (ή όποιο

περιβάλλον προγραμματισμού επιθυμείτε) τόσο για το διάβασμα του αρχείου που σας δίνεται όσο και για την κανονικοποίηση των δεδομένων εισόδου στο εύρος τιμών [-1,1]. Ακόμη θα πρέπει να αντιστοιχίσετε την κλάση του ασθενούς σε μία τιμή. Για αυτό το λόγο να αντιστοιχίσετε κάθε ασθενή, στην κατηγορία που ανήκει, σύμφωνα με τον παρακάτω πίνακα.

Car	Carcinoma	1
Fad	Fibro-adenoma	2
Mas	Mastopathy	3
Gla	Glandular	4
Con	Connective	5
Adi	Adipose	6

Η στήλη αυτή δεν χρειάζεται κανονικοποίηση.

Για την υλοποίηση αυτού του ερωτήματος χρησιμοποίηθηκε python σε Jupyter notebook στο visual studio code. Οι βιβλιοθήκες που χρησιμοποίηθηκαν είναι η pandas για την φόρτωση των δεδομένων σε ενα dataframe και απο την sklearn ο MinMaxScaler για την κανονικοποίηση. Οι άλλες δύο βιβλιοθήκες προστέθηκαν σε περίπτωση που χρειαστούν παρόλο που τελικά δεν χρειάστηκαν.

Τα imports που έγιναν για την υλοποίηση των ερωτημάτων.

```
1 import pandas as pd
2 import math
3 import numpy as np
4 from sklearn.preprocessing import MinMaxScaler
```

Έπειτα όπως προαναφέρθηκε έγινε η φόρτωση των δεδομένων στο dataframe και έπειτα ακολουθεί με μία print η πρώτη του μορφή με κάποια δείγματα.

```
[10]
1 path="/home/lefteris/Ceid/calc_theory/Project_0A_2023-24/BreastTissue.xlsx"
2
3 df_data = pd.read_excel(path, sheet_name="Data")
4
5 print(df_data)
```

```
    Case # Class          I0     PA500       HFS          DA
0        1   car   524.794072  0.187448  0.032114   228.800228  \
1        2   car   330.000000  0.226893  0.265290   121.154201
2        3   car   551.879287  0.232478  0.063530   264.804935
3        4   car   380.000000  0.240855  0.286234   137.640111
4        5   car   362.831266  0.200713  0.244346   124.912559
..     ...   ...          ...       ...       ...          ...
101    102   adi  2000.000000  0.106989  0.105418   520.222649
102    103   adi  2600.000000  0.200538  0.208043  1063.441427
103    104   adi  1600.000000  0.071908 -0.066323   436.943603
104    105   adi  2300.000000  0.045029  0.136834   185.446044
105    106   adi  2600.000000  0.069988  0.048869   745.474369
```

Έπειτα ξεκινάει η υλοποίηση του πρώτου ερωτήματος με μετατροπή των των κλάσεων στους αριθμούς που τους αντιστοιχούν με μία απλή αντικατάσταση για την κάθε κλάση και βλέπουμε ξανά το dataframe που προκύπτει.

Ερώτημα 1. Προεπεξεργασία δεδομένων.

Μετατροπή των κλάσεων στους αριθμούς που αντιστοιχούν.

```
[22]
1 df_data.loc[df_data['Class'] == 'car', 'Class'] = 1
2 df_data.loc[df_data['Class'] == 'fad', 'Class'] = 2
3 df_data.loc[df_data['Class'] == 'mas', 'Class'] = 3
4 df_data.loc[df_data['Class'] == 'gla', 'Class'] = 4
5 df_data.loc[df_data['Class'] == 'con', 'Class'] = 5
6 df_data.loc[df_data['Class'] == 'adi', 'Class'] = 6
7 print(df_data)
```

```
    Case # Class          I0     PA500       HFS          DA
0        1     1   524.794072  0.187448  0.032114   228.800228  \
1        2     1   330.000000  0.226893  0.265290   121.154201
2        3     1   551.879287  0.232478  0.063530   264.804935
3        4     1   380.000000  0.240855  0.286234   137.640111
4        5     1   362.831266  0.200713  0.244346   124.912559
..     ...   ...          ...       ...       ...          ...
101    102     6  2000.000000  0.106989  0.105418   520.222649
102    103     6  2600.000000  0.200538  0.208043  1063.441427
103    104     6  1600.000000  0.071908 -0.066323   436.943603
104    105     6  2300.000000  0.045029  0.136834   185.446044
105    106     6  2600.000000  0.069988  0.048869   745.474369
```

Τέλος χρησιμοποιούμε το εργαλείο MinMaxScaler της sklearn για να κανονικοποιήσουμε τα δεδομένα σε τιμές που ανήκουν στο σύνολο [-1,1] και εμφανίζουμε και αυτή την μορφή του dataframe.

NORMALIZE-MIN-MAX SCALER

```
1 columnsForNormalization = df_data.columns[2:]
2 print(columnsForNormalization)
3
4 scaler = MinMaxScaler(feature_range=(-1, 1))
5
6 df_data[columnsForNormalization] = scaler.fit_transform(df_data[columnsForNormalization])
7
8 print(df_data)
```

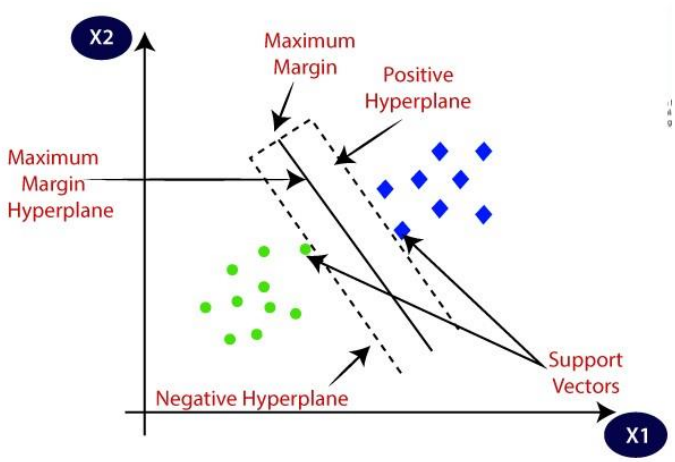
[32]

```
Index(['I0', 'PA500', 'HFS', 'DA', 'Area', 'A/DA', 'Max IP', 'DR', 'P'], dtype='object')
     Case # Class        I0     PA500       HFS        DA      Area      A/DA
0         1     1 -0.687212  0.012109 -0.631373 -0.599245 -0.922330 -0.651455  \
1         2     1 -0.831665  0.240161  0.241830 -0.805505 -0.964534 -0.698251
2         3     1 -0.667127  0.272452 -0.513725 -0.530257 -0.864481 -0.467008
3         4     1 -0.794587  0.320888  0.320261 -0.773916 -0.938860 -0.536512
4         5     1 -0.807318  0.088799  0.163399 -0.798303 -0.963075 -0.695384
..      ...   ...       ...       ...       ...       ...       ...       ...
101     102     6  0.406748 -0.453078 -0.356863 -0.040855 -0.541110 -0.071081
102     103     6  0.851687  0.087790  0.027451  1.000000  1.000000  1.000000
103     104     6  0.110122 -0.655903 -1.000000 -0.200425 -0.855686 -0.663118
104     105     6  0.629218 -0.811302 -0.239216 -0.682316 -0.942482 -0.682025
105     106     6  0.851687 -0.667003 -0.568627  0.390747 -0.543887 -0.361696
```

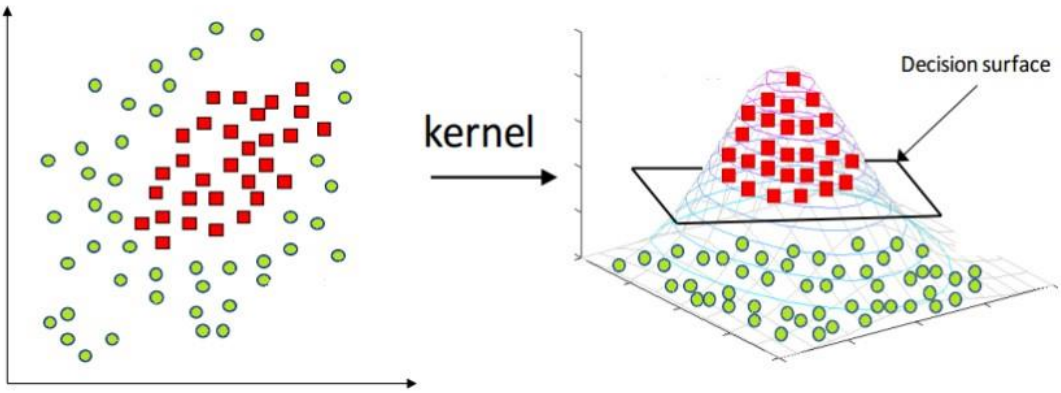
Ερώτημα 2. (20 μόρια) Να κάνετε μια σύντομη παρουσίαση του ταξινομητή Support Vector Machine και του Naive Bayes ταξινομητή. Να αντλήσετε υλικό από το διαδίκτυο και να αναφέρετε τις πηγές σας. Να κάνετε μια σύντομη σύγκριση των παραπάνω ταξινομητών με τον ΚΝΝ ταξινομητή.

Support Vector Machine:

Ο ταξινομητής support vector machine είναι ενας supervised αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται κυρίως για classification συνήθως πιο μικρών ή απλών dataset. Σκοπός αυτού του ταξινομητή είναι να βρει ένα hyperplane (υπερεπίπεδο) που να χωρίζει καλύτερα τις δύο κλάσεις. Ο SVM δουλεύει κυρίως με δύο όρους, ο πρώτος είναι τα support vectors που είναι όλα τα σημεία πάνω στις διακεκομμένες γραμμές ή ανάμεσα σε αυτές και το margin που είναι η απόσταση μεταξύ του hyperplane και των πιο κοντινών σημείων από την κάθε κλάση.



Σύμφωνα λοιπόν με svm το καλύτερο όριο μεταξύ των κλάσεων είναι εκείνο που μας δίνει την μέγιστη απόσταση από την κάθε κλάση (Maximum margin παραπάνω φωτογραφίας). Επομένως ο SVM εντοπίζει όλα τα πιθανά hyperplanes και επιλέγει εκείνο με το maximum margin. Η τεχνική αυτή μας επιφέρει μια καλή ισορροπία μεταξύ ακρίβειας και overtraining έτσι ώστε το μοντέλο να είναι εύχρηστο με διάφορα datasets. Τέλος ο SVM παρέχει ένα πολύ χρήσιμο feature, τους kernels. Οι kernels είναι συναρτήσεις οι οποίες όταν δεν είναι ευδιάκρητο το σημείο ή σημεία διαχωρισμού των δεδομένων μεταφέρει το πρόβλημα σε μεγαλύτερο επίπεδο (π.χ. απο 1D όπου τις κλάσεις διαχωρίζει ένα σημείο σε 2D όπου τις κλάσεις χωρίζει μια ευθεία γραμμή).



Naïve Bayes:

Ο Naïve bayes είναι και αυτός ενας supervised αλγόριθμος όπως ο SVM ο οποίος υποθέτει πως ένα συγκεκριμένο χαρακτηριστικό μίας κλάσης δεν έχει σχέση με κανένα άλλο χαρακτηριστικό. Είναι ιδιαίτερα αποτελεσματικός σε text classification και σε αντίθεση με τον svm (deterministic) είναι πιθανοτικός. Ο Naïve Bayes classifier χρησιμοποιεί τον κανόνα του bayes ώστε να υπολογίσει την πιθανότητα ενός input

να ανοίκει σε μια κλάση βάση της εκ των υστέρων πιθανότητα ενώς χαρακτηστικού του δεδομένης της κλάσης όπως φαίνεται στον παρακάτω τύπο.

$$P(\omega_j / x) = \frac{p(x/\omega_j)P(\omega_j)}{p(x)} = \frac{\text{πιθανοφάνεια} \times \text{εκ των προτέρων πιθ.}}{\text{αποδείξεις}}$$

Ο NB είναι απλός, γρήγορος και όταν στέκει η υπόθεση του πως τα χαρακτηριστικά είναι ανεξάρτητα μεταξύς τους έχει πολύ καλό performance συγκριτικά με άλλους αλγορίθμους. Από την άλλη εάν υπάρξουν κενά στην εκπαίδευση κάποιας κλάσης ή ενός χαραρακτηριστικού είναι πολύ πιθανό να εξάγει λάθος συμπεράσματα. Επιπλέον σε πραγματικά datasets είναι σχεδόν απίθανο να βρεθούν χαρακτηριστικά που είναι ανεξάρτητα μεταξύ τους.

Σύγκριση SVM και NB με τον KNN :

Η βασική διαφορά που διακρίνω μεταξύ του SVM και του NB με τον KNN είναι πως οι πρώτοι αλγόριθμοι εκπαιδεύουν ένα μοντέλο ταξινόμησης, δηλαδή βάση του training dataset που τους δόθηκε δημιουργούν όρια μεταξύ των κλάσεων και τα καινούρια δεδομένα ταξινομούνται βάση αυτών των ορίων στην ανάλογη κλάση. Από την άλλη ο KNN κατά την εκπαίδευση απομνυμονεύει τα δεδομένα του training dataset, έτσι συγκρίνοντας το input με τους k κοντινότερους γείτονες κατηγοριοποιεί την είσοδο. Επιπλέον ο KNN είναι πιο εύχρηστος για dataset όπου το boundary μεταξύ κλάσεων είναι δυσδιάκριτο από την άλλη ο Naïve Bayes δουλεύει μόνο για γραμμικά ελληπτικά και παραβολικά όριο ενώ ο SVM πιθανά να βρει ένα hyperplane αλλά με αρκετά μεγάλο κόστος χρόνου.

Πηγές:

SVM:https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#

NB:https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/?_gl=1*uy6cvt*_ga*MTM2ODE3MDU0MC4xNzAyNDc0MDQw*_ga_RBEPCG9FSW*MTcwMjQ3NDA0MC4xLjEuMTcwMjQ3NDA4Mi4xOC4wLjA.

KNN:https://www.datasciencecentral.com/comparing-classifiers-decision-trees-knn-naive-bayes/

Ερώτημα 3. (30 μόρια) Με χρήση της μεθόδου 5-fold cross validation και του matlab (ή όποιο περιβάλλον προγραμματισμού επιθυμείτε), εκπαιδεύστε τους παρακάτω τρεις ταξινομητές και παρουσιάστε την απόδοσή τους:

- Support Vector Machine (με Radial Basis Function σαν kernel function):
    - Ρυθμίστε την παράμετρο C με διαδοχική αναζήτηση του βέλτιστου C στο διάστημα 1-200 με βήμα 5 και χρήση γραμμικών SVM. Στη συνέχεια, ρυθμίστε την παράμετρο γ με χρήση του βέλτιστου C που βρέθηκε από πριν, και διαδοχική αναζήτηση του βέλτιστου γ στο διάστημα 0-10 με βήμα 0.5 και χρήση RBFSVM.
- Ταξινομητής KNN-K Κοντινότερου Γείτονα
    - Ρυθμίστε την παράμετρο K με διαδοχική αναζήτηση της βέλτιστης τιμής στο διάστημα 3-15.
- Αφελής Μπεϋζιανός (Naive Bayes) Ταξινομητής

Παρουσιάστε για κάθε ταξινομητή την μέση απόδοσή του με χρήση 5 fold cross validation σε σχέση με την μετρική του γεωμετρικού μέσου (Geometric Mean) της ευαισθησίας (Sensitivity) και της ειδίκευσής (Specificity) του:

Geometric Mean = sqrt (Sensitivity * Specificity)

Η μετρική αυτή χρησιμοποιείται για προβλήματα ταξινόμησης όπου παραδείγματα εκπαίδευσης της μίας κλάσης είναι περισσότερα από τα παραδείγματα εκπαίδευσης της άλλης κλάσης.

Στη συνέχεια, παρουσιάστε τα ενδιάμεσα αποτελέσματα που πήρατε από τα πειράματα για την ρύθμιση των παραμέτρων των αλγορίθμων. Γιατί η κάθε μέθοδος ταξινόμησης δίνει διαφορετικά αποτέλεσματα; Ποιά μέθοδο προτείνετε εσείς και γιατί;

Δομή Κώδικα:

Το βασικό μέρος του κώδικα βρίσκεται σε python scripts. Η κλάση του κάθε classifier καθώς και η κλάση της μετρικής geometric mean βρίσκεται στο classes.py.

classes.py imports:

Τα imports που έγιναν ήταν η βιβλιοθήκη numpy καθώς και τα απαραίτητα εργαλεία απο την βιβλιοθήκη μηχανικής μάθησης sklearn για τους classifiers και την εύρεση των βέλτιστων parameters τους.

```
import numpy as np
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

GM class:

Αρχικά δημιουργούμαι την κλάση GM για τον υπολογισμό της μετρικής. Η κλάση αυτή παίρνει σαν ορίσματα τον classifier που χρησιμοποιούμαι καθώς και τα δύο μέρη στα οποία διαχωρίζουμε το dataset, labels και features. Έπειτα δημιουργήθηκε το πρώτο function που καταμετρά τα fn,fp,tn,tp βάση ενός class label του y που

έγινε predict και του σωστού y. Ύστερα υπολογίζει το sensitivity και το specificity και τα κάνει return.

```
class GM:
    def __init__(self, estimator,X,Y):
        self.estimator = estimator
        self.X=X
        self.Y=Y
    def calculate_metrics(self, y_true, y_pred, class_label):
        true_positives = np.sum((y_true == class_label) & (y_pred == class_label))
        false_negatives = np.sum((y_true == class_label) & (y_pred != class_label))
        false_positives = np.sum((y_true != class_label) & (y_pred == class_label))
        true_negatives = np.sum((y_true != class_label) & (y_pred != class_label))

        sensitivity = true_positives / (true_positives + false_negatives) if (true_positives + false_negatives) > 0 else 0
        specificity = true_negatives / (true_negatives + false_positives) if (true_negatives + false_positives) > 0 else 0

        return sensitivity, specificity
```

Έπειτα είναι το function που κάνει split το dataset και δημιουργεί τα training και test sets για το x(features) και το y(labels) τα κάνει fit στον estimator που έχουμε δώσει ως όρισμα στην κλάση και predict για 5 διαφορετικά splits ύστερα για κάθε μία από τις 6 κλάσεις που έχουμε και καλούμε την προηγούμενη συνάρτηση με ορίσματα τα σωστά labels, αυτά που κάναμε predict και την κλάση. Υπολογίζουμε τις μετρικές και τις κάνουμε append σε 2 λίστες. Τέλος παίρνουμε 5 gmean scores ένα για κάθε κλάση και κάνουμε return την λίστα.

```
def cross_val_with_metrics(self, n_splits=5, shuffle=True, random_state=1):
    skf = StratifiedKFold(n_splits=n_splits, shuffle=shuffle, random_state=random_state)
    gm_scores = []

    for train_index, test_index in skf.split(self.X, self.Y):
        X_train, X_test = self.X.iloc[train_index], self.X.iloc[test_index]
        Y_train, Y_test = self.Y.iloc[train_index], self.Y.iloc[test_index]

        self.estimator.fit(X_train, Y_train)
        predicted_labels = self.estimator.predict(X_test)


        sensitivity_scores = []
        specificity_scores = []
        for class_label in np.unique(self.Y):
            sensitivity, specificity = self.calculate_metrics(Y_test, predicted_labels, class_label)
            sensitivity_scores.append(sensitivity)
            specificity_scores.append(specificity)

        gm_scores.append(np.sqrt(np.mean(sensitivity_scores) * np.mean(specificity_scores)))

    return gm_scores
```

CLASSIFIERS

SVM:

Στην κλάση του svm έχουμε σαν ορίσματα τα y και x και ορίζουμε 2 functions για την εύρεση των καλύτερων παραμέτρων . Η εύρεση γίνεται με χρήση της τεχνικής grid search στην οποία δημιουργείται ένα grid με τα hyperparameters έχοντας τις πιθανές τιμές από τις οποίες αναζητάμε την βέλτιστη. Έπειτα κρατάει το score της κάθε τιμής βάση της μετρικής που επιλέξαμε. Για τις ανάγκες του συγκεκριμένου προβλήματος έγινε χρήση της μετρικης f1 ή harmonic mean που παρόλο που δεν κάνει optimize το geometric mean άμεσα, αυξάνει τα true positives καθώς μειώνει τα false negatives και false positives. Η διαδικασία αυτή έγινε με cross-validation 5 επαναλήψεων. Τέλος είναι το function best_EXCE όπου γίνεται δημιουργία του classifier με τις καλύτερες παραμέτρους που βρήκαμε και καλούμε την cross_val_metrics της κλάσης GM για εκτίμηση του μοντέλου βάση του γεωμετρικού μέσου.

```
class SVM:
    def __init__(self,X,Y):
       self.best_c=0.0
       self.best_g=0.0
       self.svm=SVC(kernel='rbf')
       self.X=X
       self.Y=Y

    def best_C(self):
        find_c = {'C': list(range(1, 201, 5))}
        search = GridSearchCV(self.svm, find_c, cv=5, scoring=make_scorer(f1_score, average='weighted'))
        search.fit(self.X, self.Y)
        print("Best Parameters: ", search.best_params_)
        self.best_c=search.best_params_['C']

    def best_G(self):
        find_g = {'gamma': list(np.arange(0.0, 10.0, 0.5))}
        search = GridSearchCV(self.svm, find_g, cv=5, scoring=make_scorer(f1_score, average='weighted'))
        search.fit(self.X, self.Y)
        print("Best Parameters: ", search.best_params_)
        self.best_g=search.best_params_['gamma']

    def best_EXCE(self):
        self.best_C()
        self.best_G()

        self.svm = SVC(kernel='rbf', C=self.best_c, gamma=self.best_g)
        test=GM(self.svm,self.X,self.Y)
        geometric_mean_scores = test.cross_val_with_metrics()
        mean_value = np.mean(geometric_mean_scores)
        print("-------------THIS IS SVM-------------")
        print("Geometric Mean Scores for each fold:", geometric_mean_scores)
        print("The mean values is :", mean_value)
```

KNN:

Η ίδια ακριβώς διαδικασία γίνεται στον K-Nearest-Neighbors τόσο για την εύρεση του βέλτιστου αριθμού γειτώνων όσο και γον έλεγχο του μοντέλου με την μετρική του γεωμετρικού μέσου.

```
class KNN:
    def __init__(self,X,Y):
        self.best_k=3
        self.knn=KNeighborsClassifier(n_neighbors=self.best_k)
        self.X=X
        self.Y=Y
    def find_bk(self):
        find_k={'n_neighbors': list(range(3, 15, 1))}
        search = GridSearchCV(self.knn, find_k, cv=5, scoring=make_scorer(f1_score, average='weighted'))
        search.fit(self.X, self.Y)
        print("Best Parameters: ", search.best_params_)
        self.best_k=search.best_params_['n_neighbors']
    def best_EXCE(self):
        self.find_bk()
        self.knn = KNeighborsClassifier(n_neighbors=self.best_k)

        test=GM(self.knn,self.X,self.Y)
        geometric_mean_scores = test.cross_val_with_metrics()
        mean_value = np.mean(geometric_mean_scores)
        print("-------------THIS IS KNN-------------")
        print("Geometric Mean Scores for each fold:", geometric_mean_scores)
        print("The mean values is :", mean_value)
```

NB:

Στον Naïve Bayes εκτελούμε επιτόπου το μοντέλου και το αξιολογούμε αφού δεν χρειάζεται να βρούμε κάποια βέλτιστη παράμετρο.

```
class NB:
    def __init__(self,X,Y):
        self.nb = GaussianNB()
        self.X=X
        self.Y=Y
    def best_EXCE(self):
        test=GM(self.nb,self.X,self.Y)
        geometric_mean_scores = test.cross_val_with_metrics()
        mean_value = np.mean(geometric_mean_scores)
        print("-------------THIS IS NAIVE BAYES-------------")
        print("Geometric Mean Scores for each fold:", geometric_mean_scores)
        print("The mean values is :", mean_value)
```

ΑΠΟΤΕΛΕΣΜΑΤΑ-ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ:

Αρχικά δημιουργήθηκε ένα script create_dataset με τον κώδικα των πρώτων ερωτημάτων για δημιουργία του dataset και διαχωρισμός του σε X(features) και Y(classes).

```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

def create_dataset():
    path="/home/lefteris/Ceid/calc_theory/Project_0A_2023-24/BreastTissue.xlsx"

    df_data = pd.read_excel(path, sheet_name="Data")

    df_data.loc[df_data['Class'] == 'car', 'Class'] = 1
    df_data.loc[df_data['Class'] == 'fad', 'Class'] = 2
    df_data.loc[df_data['Class'] == 'mas', 'Class'] = 3
    df_data.loc[df_data['Class'] == 'gla', 'Class'] = 4
    df_data.loc[df_data['Class'] == 'con', 'Class'] = 5
    df_data.loc[df_data['Class'] == 'adi', 'Class'] = 6

    df_data['Class'] = df_data['Class'].astype(int)

    columnsForNormalization = df_data.columns[2:]

    scaler = MinMaxScaler(feature_range=(-1, 1))

    df_data[columnsForNormalization] = scaler.fit_transform(df_data[columnsForNormalization])

    return df_data

def create_X_Y(dataset):
    X = dataset.drop(['Case #', 'Class'], axis=1)
    Y = dataset['Class']
    return X,Y
```

Το script του ερωτήματος 3, erotima3.py τρέχει τα μοντέλα και εμφανίζει τα αποτελέσματα της αξιολόγησης βάσης της geometric mean μετρικής.

```
from create_dataset import create_dataset,create_X_Y
from classes import SVM,KNN,NB

df_data=create_dataset()
X,Y=create_X_Y(df_data)



print("(ΕΡΩΤΗΜΑ 3)")
print("\n\n")
svm=SVM(X,Y)
svm.best_EXCE()
print("\n\n")
knn=KNN(X,Y)
knn.best_EXCE()
print("\n\n")
nb=NB(X,Y)
nb.best_EXCE()
print("\n\n")
```

RESULTS:

```
(ΕΡΩΤΗΜΑ 3)



Best Parameters:  {'C': 26}
Best Parameters:  {'gamma': 3.5}
-------------THIS IS SVM-------------
Geometric Mean Scores for each fold: [0.7693049235247862, 0.7889026971082811, 0.7598302757394464, 0.8607684327645058, 0.7758886176106641]
The mean values is : 0.7909389893495367



Best Parameters:  {'n_neighbors': 4}
-------------THIS IS KNN-------------
Geometric Mean Scores for each fold: [0.8423114570976348, 0.7513402855184483, 0.8802514715185809, 0.7975429770805373, 0.7761128304798467]
The mean values is : 0.8095118043390096



-------------THIS IS NAIVE BAYES-------------
Geometric Mean Scores for each fold: [0.6927599107937622, 0.7602781568722026, 0.7889026971082811, 0.7308124129086019, 0.8418552905561372]
The mean values is : 0.7629216936477969
```

Όπως παρατηρούμε τα καλύτερα αποτελέσματα δίνει ο SVM και ο KNN με τον k-nearest-neighbor να είναι λίγο καλύτερος. Ο Naïve Bayes όπως ήταν αναμενόμενο είναι ο χειρότερος αφού προκειμένου να δώσει βέλτιστα αποτέλεσμα προυποθέτει κάθε χαρακτηρηστικό που αξιολογούμε να είναι τελείως ανεξάρτητο από τα υπόλοιπα και άρα να μιλάμε για πολύ διακριτές κλάσεις πράγμα που δεν ισχύει στο συγκεκριμένο dataset. Από την άλλη τα αποτελέσματα του SVM και του KNN είναι

πολύ κοντά, από την μία έχουμε ένα μικρό dataset και κλάσεις που είναι παρόμοιες μεταξύ τους που ευνοεί τον KNN αλλά από την άλλη έχουμε πολλά features και άρα πολλές διαστάσεις για να δουλέψει ο svm. Εν τέλη ενώ έχουμε παρόμοια αποτελέσματα ο KNN είναι πιο εύχρηστος στην προκειμένη περίπτωση αφού έχει και μικρότερο κόστος από τον svm, δίνει λίγο καλύτερα αποτελέσματα και δουλεύει καλύτερα για μικρό αριθμό χαρακτηριστηκών που θα ευνοήσει το τελευταίο ερώτημα.

Ερώτημα 4 (10 μόρια). Να κάνετε μια σύντομη παρουσίαση της μεθόδου Student t-test. Να αντλήσετε υλικό από το διαδίκτυο και να αναφέρετε τις πηγές σας.

student's T-Test:

Η μέθοδος student's T-Test είναι ένα στατιστικό hypothesis test το οποίο ελέγχει εάν δύο δείγματα του dataset ανήκουν στο ίδιο population. Αυτό που κάνει αυτή η μέθοδος είναι να συγκρίνει τους μέσους όρους των δύο αυτών δειγμάτων και να δει αν διαφέρουν σημαντικά. Η μέθοδος αυτή χρησιμοποιείται σε dataset με άγνωστο variance που συνήθως ακολουθεί κανονική κατανομή, μια υπόθεση ότι τα δείγματα διαφέρουν καθώς και την null hypothesis που είναι τα δείγματα να μην διαφέρουν και να ανήκουν στον ίδιο πληθυσμό. Ο σημαντικότερος όρος αυτής της μεθόδου είναι το T-statistic το οποίο συμβολίζει την διαφορά μεταξύ των μέσων όρων των δειγμάτων και υπολογίζεται από τον παρακάτω τύπο .

$$t = (\bar{x} - \mu) / (\sigma / \sqrt{n})$$

Where,

- ***t** is the t-value,*
- ***$\bar{x}$** is the Sample mean,*
- ***μ** is the Population mean,*
- ***σ** is the Sample standard deviation, and*
- ***n** is the Sample size.*

Εξίσου σημαντικός όρος είναι το a που συμβολίζει την πιθανότητα να απορριφθεί το null hypothesis. Τέλος είναι το p-value που είναι η πιθανότητα να βρούμε το t-statistic που υπολογίστηκε δεδομένου ότι το null hypothesis ισχύει. Δηλαδή ένα μικρό p-value σημαίνει ότι η διαφορά μεταξύ των δειγμάτων έχει μικρή πιθανότητα να παρατηρείται τυχαία και άρα δεν ισχύει το null hypothesis.

Το Test αυτό ξεκινά θέτοτας την hypothesis και την null hypothesis. Έπειτα ξεκινά να υπολογίζει το t-statistic και το p-value. Συγκρίνει το p-value με το a (συνήθως είναι είται 0.05 είτε 0.01). Αν το p-value είναι μικρότερο απο το A απορίπτουμε το null hypothesis και άρα συμπαιρένουμε ότι τα δείγματα δεν ανήκουν στο ίδιο population ειδάλλως το null hypothesis ισχύει και ανήκουν στο ίδιο.

Υπάρχουν τρία είδη t-test:

Independent samples t-test: Συγκρίνει τα data 2 ομάδων για να βρει αν οι διαφορές τους είναι σημαντικές ή τυχαίες. Αυτή η μέθοδος χρησιμοποιείται για διαφορετικά δείγματα για τα οποία δεν έχουμε πληροφορίες για τον πληθυσμό τους.

One sample t-test: Συγκρίνει τα data μιας ομάδας με ένα γνωστό μέσο όρο. Χρησιμοποιείται κυρίως για μικρά δείγματα, κάτω από 30 τυχαία δεδομένα.

Paired sample t-test: Συγκρίνει data απο δύο ομάδες και ταιριάζει δείγματα από το ένα group με δείγματα από το άλλο για να δει αν οι μέσοι όροι είναι ίδιοι.Χρησιμοποιείται σε αλληλοεξαρτόμενα/παρόμοια δείγματα και σκοπός της μεθόδου είναι να δει αν η διαφορά των μέσων όρων τους είναι 0.

Ερώτημα 5. (30 μόρια) Για τα δεδομένα που σας δίνονται διατάξτε τα χαρακτηριστικά εισόδου με χρήση της μεθόδου student t-test με βάση την σημαντικότητά τους στην πρόβλεψη του καρκίνου του μαστού. Στη συνέχεια, κρατείστε τα 4 πιο σημαντικά χαρακτηριστικά, και επαναλάβετε την εκπαίδευση του βέλτιστου ταξινομητή που βρήκατε από προηγούμενο ερώτημα. Σχολιάστε την τελική απόδοση.

Επιλογή σημαντικότερων χαρακτηριστικών με Independent samples t-test της βιβλιοθήκης scipy. Στον παρακάτω κώδικα διαχωρίζουμε το dataset μας σε target_class (class 1) και other classes δημιουργώντας δύο διαφορετικούς πληθυσμούς. Έπειτα υπολογίζουμε το t_statistic και το p_value . Συγκρίνουμε το p με a=0.05 και δημιουργείται μία λίστα με τα p_values απο το μικρότερο στο μεγαλύτερο. Τέλος δημιουργούμε δύο νέες λίστες, μία με τα χαρακτηριστικά που θέλουμε να κρατήσουμε και μία με αυτά που δεν θέλουμε.

```python
from scipy.stats import ttest_ind
from create_dataset import create_dataset,create_X_Y
from classes import KNN

df_data=create_dataset()
X,Y=create_X_Y(df_data)

target_class = df_data[df_data['Class'] == 1]
other_classes = df_data[df_data['Class'] != 1]
features = list(df_data.columns)
features.remove('Case #')
features.remove('Class')
p_values=[]

for feature in features:
    t_statistic, p_value = ttest_ind(target_class[feature], other_classes[feature])
    p_values.append([feature,p_value])
    if p_value < 0.05:
        print(f"Feature '{feature}' is significant between class_1 and other classes.")
    else:
        print(f"Feature '{feature}' is not significant between class_1 ")

sorted_p_value_list = sorted(p_values, key=lambda x: x[1])
sorted_p_value_list = [item[0] for item in sorted_p_value_list]
i_want_these=sorted_p_value_list[:4]
get_rid_of_these=sorted_p_value_list[4:]
```

Αποτελέσματα χαρακτηριστικών:

```
Feature 'I0' is significant between class_1 and other classes.
Feature 'PA500' is significant between class_1 and other classes.
Feature 'HFS' is significant between class_1 and other classes.
Feature 'DA' is not significant between class_1
Feature 'Area' is not significant between class_1
Feature 'A/DA' is not significant between class_1
Feature 'Max IP' is not significant between class_1
Feature 'DR' is not significant between class_1
Feature 'P' is significant between class_1 and other classes.
```

Έπειτα βγάζουμε από το dataset τα 4 features που δεν μας ενδιαφέρουν και δημιουργούμε τα νέα Xnew και Ynew. Δημιουργούμε ένα νέο αντικείμενο KNN και το αξιολογούμε.

```
print('\n')
print('ΕΡΩΤΗΜΑ 5)')
print('\n\n')
print('THIS IS THE LIST WITH THE 4 MORE IMPORTANT FEATURES')
print(i_want_these)
print('\n\n')
new_dataset = df_data.drop(get_rid_of_these, axis=1)
Xnew,Ynew = create_X_Y(new_dataset)
knn_new=KNN(Xnew,Ynew)
knn_new.best_EXCE()
```

ΑΠΟΤΕΛΕΣΜΑΤΑ:

```
ΕΡΩΤΗΜΑ 5)


THIS IS THE LIST WITH THE 4 MORE IMPORTANT FEATURES
['PA500', 'HFS', 'I0', 'P']


Best Parameters:  {'n_neighbors': 6}
-------------THIS IS KNN-------------
Geometric Mean Scores for each fold: [0.7701540462154054, 0.7598302757394464, 0.8802514715185809, 0.7607257743127307, 0.7291128468570203]
The mean values is : 0.7800148829286367
```

Όπως βλέπουμε η συνολική απόδοση του αλγορίθμου μειώνεται αυτό οφείλεται στο γεγονός ότι ο classifier αξιολογείται για το classifying 6 διαφορετικών κλάσεων και εμείς κρατήσαμε τα τέσσερα χαρακτηριστικά που ευνοούν την ταξινόμηση της πρώτης κλάσης. Αυτό που καταφέραμε ωστόσο είναι να βελτιστοποιήσουμε την ικάνότητα του K-NN να εντοπίζει περισταστικά καρκίνου του μαστού (class 1) μειώνοντας βέβαια την ικανότητα του να ταξινομεί τις υπόλοιπες κλάσεις. Επιπλέον πρέπει να σημειωθεί ότι οποιοσδήποτε άλλος ταξινομητής θα παρουσίαζε ακόμα μικρότερη απόδοση. Αυτό οφείλεται στην ικανότητα του KNN να αποδίδει καλύτερα σε datasets με λίχα features είτε datasets στα οποία κρατάμε συγκεκριμένα χαρακτηριστικά.

ΠΑΡΑΤΗΡΗΣΕΙΣ:

- Η εργασία αποτελείται από 2 μέρη. Στο 1^{ο} μέρος θα πρέπει να απαντήσετε τα ερωτήματα 1 και 2. Η αναφορά του 1^{ου} μέρους της εργασίας πρέπει να αναρτηθεί στη σελίδα του μαθήματος στο e-class, στις 17/12/2023 μέχρι τις 23.55.
- Η αναφορά του 2^{ου} μέρους της εργασίας πρέπει να αναρτηθεί στη σελίδα του μαθήματος στο e-class την παραμονή της εξέτασης του μαθήματος, μέχρι τις 23.55 (μετά από αυτή την ώρα το σύστημα θα κλείσει).

- Για την αναφορά της εργασίας σας, θα χρησιμοποιήσετε το αρχείο της εκφώνησης. Μετά από κάθε υποερώτημα θα έχετε την απάντησή σας, στην οποία θα περιγράφετε τη μεθοδολογία, τα εργαλεία που χρησιμοποιήσατε κλπ. Στα ερωτήματα που αναπτύσσετε κώδικα, θα πρέπει να έχετε κάποια screen shots από τα τρεξίματα και σε χωριστό αρχείο τον κώδικα. Το όνομα του αρχείου με τον κώδικα, θα πρέπει να περιέχει τον αριθμό του αντίστοιχου υποερωτηματος.
- Θα αναρτήσετε ένα .zip αρχείο που θα περιέχει την αναφορά και τον κώδικα. Στο όνομα του αρχείου πρέπει να υπάρχει το επώνυμό σας και το αρχικό γράμμα του ονόματός σας και ο ΑΜ.