ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΜΕΡΟΣ Α’ 2021-2022

ΣΠΕΝΤΖΑΡΗΣ ΠΑΝΑΓΙΩΤΗΣ 1071110

Θεωρία αποφάσεων

Link κωδικα: https://github.com/Pspetz/Liver_disease_prediction

Στο συγκεκριμένο repository ο κώδικας για το μέρος Α της εργασίας είναι στο liver_disease_part_A.py ενώ η συνολική εργασία(μέρος Α+μέρος Β) περιέχεται στο Liver_disease.ipynb

Ερώτημα 1

1. ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΚΑΙ ΠΡΟΕΤΟΙΜΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Ένα από τα σημαντικότερα βήματα στον τομέα της μηχανικής μάθησης είναι η σωστή προ επεξεργασία τον δεδομένων. Στην άσκηση μας ζητείται να αναφέρουμε πόσα χαρακτηριστικά και δείγματα εκπαίδευσης περιέχει το dataset που καλούμαστε να χειριστούμε . Έπειτα να αντιστοιχήσουμε την στήλη που περιέχει το φύλο του ανθρώπου σε δυαδικές τιμές (0 και 1) και τέλος να κανονικοποιήσουμε τα δεδομένα.

Πιο συγκεκριμένα πριν ξεκινήσουμε την επεξεργασία των δεδομένων πρέπει να μελετήσουμε το σύνολο των δεδομένων που καλούμαστε να διαχειριστούμε .Βλέπουμε ότι το dataset περιέχει **11 δείγματα(χαρακτηριστικά)** όπου συνολικά έχουμε **583 δείγματα εκπαίδευσης** .Αρχικά

διαβάζουμε το dataset χρησιμοποιώντας την βιβλιοθήκη pandas , έπειτα δίνουμε στις στήλες κατάλληλες ονομασίες με βάση το ILPD και στην συνέχεια βλέπουμε τις πληροφορίες του συνόλου δεδομένων ώστε να δούμε αν υπάρχουν στήλες όπου περιέχουν NaN values. Στην συγκεκριμένη περίπτωση παρατηρούμε ότι η στήλη Albumin_Globulin έχει σε 4 γραμμές NaN πράγμα που δημιουργεί πρόβλημα καθώς δεν επιτρέπεται κάποια στήλη να μην περιέχει τιμή αναφοράς.

```
csv = pd.read_csv("Indian Liver Patient Dataset (ILPD).csv",header=None)
```

```
csv.columns=["Age",'Gender','Tb_Bilirubin','DB_Bilirubin','Alkaline_Phosphotase','sgpt ','Sgot','Protiens','Albumin','Albumin_Globulin','Disease']

#Missing Values
csv.info()
csv.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Age                  583 non-null    int64  
 1   Gender               583 non-null    object 
 2   Tb_Bilirubin         583 non-null    float64
 3   DB_Bilirubin         583 non-null    float64
 4   Alkaline_Phosphotase 583 non-null    int64  
 5   sgpt                 583 non-null    int64  
 6   Sgot                 583 non-null    int64  
 7   Protiens             583 non-null    float64
 8   Albumin              583 non-null    float64
 9   Albumin_Globulin     579 non-null    float64
 10  Disease              583 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

Υπάρχουν 2 περιπτώσεις:
→ Εάν το σύνολο με τα δεδομένα είναι μεγάλο μπορούμε να αφαιρέσουμε τις σειρές με τις τιμές που λείπουν και να χρησιμοποιήσουμε τα υπόλοιπα δεδομένα για να προβλέψουμε τις τιμές.
→ Όταν όμως έχουμε μικρότερα σύνολα δεδομένων μπορούμε να αντικαταστήσουμε τις τιμές αυτές που λείπουν με τον μέσο όρο των υπολοίπων δεδομένων της στήλης αυτής.

```
csv['Albumin_Globulin'] = csv['Albumin_Globulin'].fillna(csv['Albumin_Globulin'].mean())
csv.info()
csv.isna().sum()
csv
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Age                  583 non-null    int64  
 1   Gender               583 non-null    object 
 2   Tb_Bilirubin         583 non-null    float64
 3   DB_Bilirubin         583 non-null    float64
 4   Alkaline_Phosphotase 583 non-null    int64  
 5   sgpt                 583 non-null    int64  
 6   Sgot                 583 non-null    int64  
 7   Protiens             583 non-null    float64
 8   Albumin              583 non-null    float64
 9   Albumin_Globulin     583 non-null    float64
 10  Disease              583 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

Στην συγκριμένη περίπτωση αντικατέστησα τα NaN με το μέσο όρο των υπολοίπων της συγκεκριμένης στήλης όπως φαίνεται και παρακάτω στην στήλη ώστε να μην υπάρχει πεδίο χωρίς τιμή.

Στην συνέχεια χρησιμοποιώντας regex μέσω της pandas αντικατέστησα στην στήλη Gender όσα field περιέχουν το male με 0 ενώ τα female με 1.Αυτό το κάνουμε ώστε να προσδιορίσουμε το φύλο με δυαδική ταξινόμηση καθώς με string δεν μπορούμε να αξιοποιήσουμε την συγκριμένη στήλη.

```
#Replace Female with 1 and Male with 0
csv=csv.replace(regex=['Female'],value='1')
csv=csv.replace(regex=['Male'],value='0')

csv
```

	Age	Gender	Tb_Bilirubin	DB_Bilirubin	Alkaline_Phosphotase	sgpt	Sgot	Protiens	Albumin	Albumin_Globulin	Disease
0	65	1	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	0	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	0	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	0	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	0	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...	...	...
578	60	0	0.5	0.1	500	20	34	5.9	1.6	0.37	2
579	40	0	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	0	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	0	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	0	1.0	0.3	216	21	24	7.3	4.4	1.50	2

583 rows × 11 columns

Έπειτα ελέγχουμε την ακολουθία των στοιχείων ώστε να βεβαιωθούμε ότι έχουμε την σωστή αντιστοιχία μεταξύ του string και της δυαδικής τιμής που έχει ανατεθεί.

```
#Replace Female with 1 and Male with 0
csv=csv.replace(regex=['Female'],value='1')
csv=csv.replace(regex=['Male'],value='0')

csv['Gender'].sum()
```

✓ 0.3s Python

```
'1000001100000100001100000000010001101100000010001011000001001000000000111001010110000000000000000000010000001000000000001000100110001000010010100000100000000000001000100000000000000000100000001010010000011000100000000000011000001001100011000000000110000000010001000000000010000001010100000011101101100110101000000110000100000011100000100000001001110000101010000111001100110001100000000010000010100100000100000001100010001011011100011111000111000010000110011110001000000000000110001010000000000000000100000000000100100001001000101000011010100000000000010101000000000000'
```

2.ΠΡΟΣΑΡΜΟΓΗ ΔΕΔΟΜΕΝΩΝ ΣΕ ΔΙΑΦΟΡΕΤΙΚΗ ΚΛΙΜΑΚΑ

Επεξήγηση:

Normalization: Όταν εφαρμόσουμε την τεχνική του Normalization στα samples που διαθέτουμε, μετατοπίζουμε πρώτα την κλίμακα έτσι ώστε να ξεκινά από το 0 και στη συνέχεια τη συμπιέζουμε ώστε να τελειώσει στο 1. Αυτό γίνεται αφαιρώντας πρώτα την ελάχιστη τιμή και μετά διαιρώντας

με τη νέα μέγιστη τιμή (η οποία είναι η παλιά μέγιστη τιμή μείον την παλιά ελάχιστη τιμή).Άρα καταλήγουμε όλα τα δείγματα μας να ανήκουν στην κλίμακα [0,1].

Standardize: Όταν εφαρμόσουμε την τεχνική του Standarization στα samples που διαθέτουμε, τότε κάθε μεταβλητή θα έχει μέσο όρο 0 και τυπική απόκλιση 1. Για να επιτευχθεί αυτό αρχικά αφαιρούμε τον μέσο όρο για να κεντράρουμε τη μεταβλητή και μετά διαιρούμε με την τυπική απόκλιση. Με αυτόν τον τρόπο η τυπική απόκλιση θα είναι 1.

Centering: Όταν εφαρμόσουμε την μέθοδο του Centering τότε αφαιρούμε τον μέσο όρο κάθε μεταβλητής ώστε ό μέσος όρος τελικά να μηδενιστεί. Επειδή όμως στο συγκεκριμένο project έχουμε εφαρμόσει την τεχνική του Bow και έχουμε την συχνότητα εμφάνισης κάθε λέξης δεν θα χρησιμοποιήσουμε centering γιατί δεν θα βοηθήσει τα δεδομένα μας.

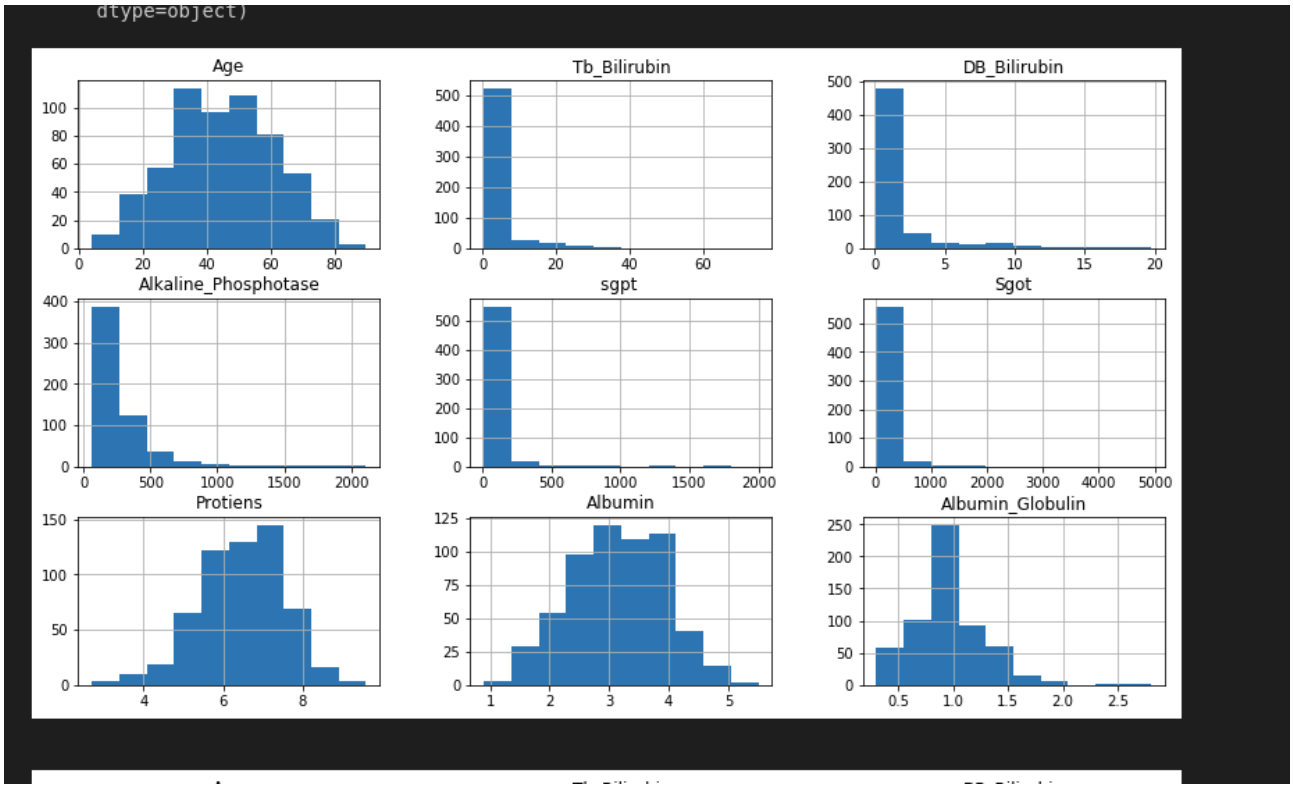
Στην άσκηση μας ζητείτε τα δεδομένα να κανονικοποιηθούν στο εύρος [-1,1], με αυτό τον τρόπο πετυχαίνουμε να έχουμε μεγαλύτερη ποικιλία χαρακτηριστικών σε μια πιο περιορισμένη κλίμακα.
Αρχικά έχω εφαρμόσει την τεχνική train_test_split ώστε να διαχωρίσω τα δεδομένα σε 75% σύνολο εκπαίδευσης και 25% σε δεδομένα testing ώστε να αξιοποιηθούν ως προς έλεγχο για το τελικό μοντέλο που θα επιλέξουμε ως το καταλληλότερο.
Έπειτα μέσω της keras με χρήση του MinMaxscaler κανονικοποιούμε τα δεδομένα από (-1,1) όπως ζητήθηκε.
Τέλος τυπώνω το shape των δεδομένων ώστε να γνωρίζουμε πόσα δεδομένα εκπαίδευσης και testing έχουμε.

```
#split dataset to train and test data
X_train,X_test,Y_train,Y_test =train_test_split(X_data,Y_data,test_size=0.25)

def preprocessing(X_train,Y_train,X_test,Y_test,type="MinMax"):

        #NORMALIZATION#
    if type == "Normalization":
        X_train_normalized = tf.keras.utils.normalize(X_train)
        X_test_normalized = tf.keras.utils.normalize(X_test)
        return X_train_normalized,Y_train,X_test_normalized,Y_test

        #NORM-WITH MINMAX#
    elif type == "MinMax":
        scaler = MinMaxScaler(feature_range=(-1, 1))
        X_train_minmax = scaler.fit_transform(X_train)
        X_test_minmax = scaler.fit_transform(X_test)
        return X_train_minmax , Y_train ,X_test_minmax ,Y_test

X_train_minmax,Y_train,X_test_minmax,Y_test=preprocessing(X_train,Y_train,X_test,Y_test,type="MinMax")
print(X_train_minmax.shape,X_test_minmax.shape,Y_train.shape,Y_test.shape)
```

```
(437, 10) (146, 10) (437,) (146,)
```

3.ΔΙΟΡΘΩΣΗ ΛΟΞΗΣ ΚΑΤΑΝΟΜΗΣ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Τέλος κάνοντας plot τα δεδομένα κάθε στήλης ελέγχουμε αν υπάρχει στήλη **όπου τα δεδομένα της έχουν λοξή κατανομή** πράγμα που θα οδηγήσει το μοντέλο να εκπαιδευτεί σε πολύ μεγαλύτερο αριθμό δεδομένων που βρίσκονται σε μια συγκεκριμένη περιοχή και θα είναι λιγότερο πιθανό να μπορεί να προβλέψει με επιτυχία τιμές που βρίσκονται μακριά από την περιοχή αυτή. Έτσι οι ακραίες τιμές πρέπει να αντιμετωπιστούν ώστε να μειώσουμε το φαινόμενο αυτό. Στην συγκριμένη περίπτωση μέσω της μηδενικής υπόθεσης και του p value.



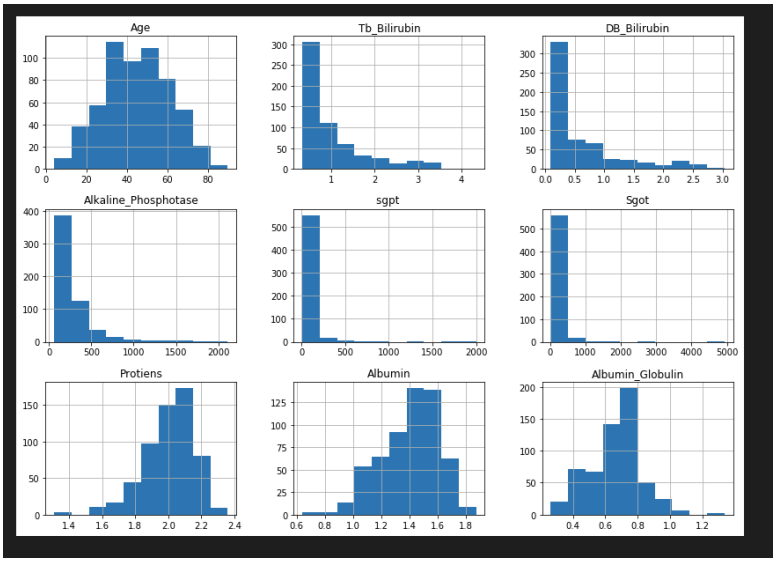
Η p value χρησιμοποιείται για να επικύρωση μια υπόθεση έναντι των δεδομένων που παρατηρούμε. Όσο μικρότερη η p τόσο μεγαλύτερη διαφορά έχουμε στην υπόθεση που θεωρήσαμε.
P <=0.05 κατά της μηδενικής υπόθεσης
P >0.05 αποδοχή μηδενικής υπόθεσης
P = 0.05 οριακή απόφαση καθώς μπορούν να ισχύει ένα από τα δύο οριακά.
Βγάζουμε το συμπέρασμα ότι οι στήλες Albumin,DB_Bilirubin,Tb_Bilirubin,Albumin_Globulin,Protiens **εμφανίζουν λοξή κατανομή** των δεδομένων για αυτό χρησιμοποιούμε την τεχνική του log transformation.

```
#1=> Liver patience
#csv['Disease'].replace(to_replace = 1, value = 0, inplace=True)
#csv['Disease'].replace(to_replace = 2, value = 1, inplace=True)

csv

from scipy.stats import shapiro
#Age
resp_age,resp_Tb_Bilirubin=csv.Age,csv.Tb_Bilirubin
shapiro(resp_Tb_Bilirubin)[1]
#Gender
```

```
.5773978206485132e-38
```

```
#Create Input && output data
X_data = csv.drop('Disease',axis=1)
Y_data = csv['Disease']

# Visualize skewed continuous features of original data

X_data.hist(figsize=(14,8))


# Skewed features are Albumin, Direct Bilirubin, A/G ratio, Tota Bilirubin, Total Protein
#Log-transform the skewed features but also include zero values)
skewed = ['Albumin', 'DB_Bilirubin', 'Tb_Bilirubin', 'Albumin_Globulin', 'Protiens']
X_data[skewed] = X_data[skewed].apply(lambda x: np.log(x + 1))
X_data[skewed] = X_data[skewed]
# Visualize the new log distributions
X_data.hist(figsize=(14,10))
```

Όπως παρατηρούμε ο μετασχηματισμός αυτός βοήθησε να διορθώσουμε την λοξή κατανομή των δεδομένων και η νέα κατανομή να φαίνεται πιο φέρνει πιο κοντά στην κανονικοποίηση.



Ερώτημα 2

Μάθηση κατά Bayes - Αφελής Ταξινομητής Bayes (Naive Bayes)

Στη μάθηση κατά Bayes κάθε παράδειγμα εκπαίδευσης μπορεί σταδιακά να μειώνει ή να αυξάνει την πιθανότητα να είναι σωστή μια υπόθεση. Η δυσκολία στην εφαρμογή του θεωρήματος του Bayes εγκείται στην γνώση πολλών πιθανών τιμών πιθανότητας, όταν δεν μπορούν να υπολογιστούν και να εκτιμηθούν επακριβώς από προηγούμενες περιπτώσεις και προηγούμενη γνώση.Η δυσκολία που σχετίζεται με αυτό οδήγησε σε μια αφελή προσέγγιση Bayes , η οποία βασίζεται σε μια απλή πεποίθηση ότι οι μεταβλητές είναι ανεξάρτητες η μία από την άλλη. Τα τελευταία χρόνια, η μελέτη των πιθανοτικών μοντέλων έχει γίνει ένας δημοφιλής τομέας έρευνας. Ένας από τους πιο σημαντικούς στόχους αυτής της προσέγγισης της πιθανότητας είναι η εύρεση μιας πιο πιθανής υπόθεσης για έναν δεδομένο χώρο υποθέσεων. Αυτό μπορεί να επιτευχθεί αξιοποιώντας εκ των προτέρων γνώσεις και υποθέσεις σχετικά με αυτόν τον χώρο Η.
Η συνάρτηση πιθανότητας L(h) με δεδομένη την υπόθεση h ανήκει στο Η, το οποίο είναι υποσύνολο του D, μπορεί να υπολογιστεί χρησιμοποιώντας το θεώρημα του Bayes.

$$\Pr(h \mid D) = \frac{\Pr(h)\Pr(D \mid h)}{\Pr(D)}$$

Pr(D | h) : η δεσμευμένη πιθανότητα που εκφράζει το ενδεχόμενο παρατήρησης των δεδομένων του D, ισχυούσης της h (πιθανοφάνεια –likelihood).
Pr(D) : η εκ των προτέρων πιθανότητα παρατήρησης των δεδομένων του D. (Ο συγκεκριμένος όρος απλοποιείται και δε συμμετέχει στους υπολογισμούς).
Pr(h | D) : η ζητούμενη εκ των υστέρων πιθανότητα ισχύος της h δεδομένης της παρατήρησης των δεδομένων του D.
Η αναζήτηση της πιο πιθανής υπόθεσης h δεδομένου του D ανάγεται στην εύρεση της υπόθεσης με τη μεγαλύτερη εκ των υστέρων πιθανότητα (maximum a-posteriori ή MAP hypothesis). Ορίζεται η υπόθεση αυτή ως εξής:

$$h_{MAP} = \arg\max_{h \in H}\{\Pr(h \mid D)\} = \arg\max_{h \in H}\left\{\frac{\Pr(h)\Pr(D \mid h)}{\Pr(D)}\right\} = \arg\max_{h \in H}\{\Pr(h)\Pr(D \mid h)\}$$

Ο παραπάνω αλγόριθμος υπάρχει στη βιβλιογραφία με το όνομα Βέλτιστος Ταξινομητής Bayes και αποδεικνύεται θεωρητικά πως είναι σε θέση να υπολογίσει το άνω φράγμα των επιδόσεων ενός συστήματος ταξινόμησης για ένα συγκεκριμένο πρόβλημα.

Για το πρόβλημα της ταξινόμησης, χρησιμοποιώντας διανυσματική αναπαράσταση των δεδομένων, γίνονται οι παρακάτω υποθέσεις:

C: τυχαία μεταβλητή που δείχνει την κλάση ενός στιγμιότυπου.
X: διάνυσμα τυχαίων μεταβλητών που δείχνει τις τιμές των παρατηρούμενων χαρακτηριστικών.
c: μια συγκεκριμένη ετικέτα κλάσης.
x: ένα συγκεκριμένο παρατηρούμενο διάνυσμα.

Αφελής Ταξινομητής Bayes (naive Bayes classifier)

Το θεώρημα του Bayes είναι ένα βασικό εργαλείο για την κατανόηση πολλών τύπων δεδομένων, συμπεριλαμβανομένου του κειμένου. Μας δίνει τη δυνατότητα να κάνουμε προβλέψεις με βάση τις πιθανότητες διαφόρων συνθηκών.Ωστόσο, η δυσκολία στην εφαρμογή μοντέλων Bayes είναι ότι απαιτούν υποθέσεις σχετικά με όλους τους πιθανούς συνδυασμούς και μη συνδυασμούς χαρακτηριστικών. Αυτό σημαίνει ότι, για να λειτουργεί καλά ένα μοντέλο, πρέπει να μπορούμε να χρησιμοποιούμε ανεξάρτητες μεταβλητές ως βάση για πρόβλεψή μας.
Αυτό ξεπερνάτε με τη χρήση του απλού κατηγοριοποιητή Bayes (simple/naive Bayes classifier), στον οποίο γίνεται παραδοχή ότι τα χαρακτηριστικά που χρησιμοποιούνται ως τιμές εισόδου είναι ανεξάρτητα μεταξύ τους. Ο απλός ταξινομητής Bayes είναι μια πρακτική μέθοδος μάθησης που στηρίζεται σε στατιστικά στοιχεία (κατανομές πιθανότητας) και προσφέρει μια απλή πιθανοθεωρητική προσέγγιση στα προβλήματα μάθησης με επίβλεψη, όπου ο στόχος είναι να προβλεφθεί με ακρίβεια η κλάση των στιγμιοτύπων δεδομένων, χρησιμοποιώντας μοντέλα μηχανικής μάθησης που λαμβάνουν υπόψη τις πληροφορίες από το εκπαιδευτικό σύνολο που ανήκει σε αυτήν την τάξη. Ο ταξινομητής Naive Bayes βασίζεται σε δυο σημαντικές υποθέσεις, ενώ παράλληλα υποθέτει ότι κάθε χαρακτηριστικό των στιγμιότυπων είναι στοχαστικά ανεξάρτητο των υπόλοιπων, δεδομένης της κλάσης και του γεγονότος ότι δεν υπάρχουν άλλα κρυφά χαρακτηριστικά που να επηρεάζουν την διαδικασία της πρόβλεψης. Έτσι η πιθανότητα της σχέσης μετατρέπεται σε γινόμενο πιθανοτήτων, όπως φαίνεται παρακάτω:

$$\arg\max_{\forall c}\{p(C=c \mid \mathrm{X}=x)\} = \arg\max_{\forall c}\{p(C=c)\prod_i p(X_i=x_i \mid C=c)\}$$

Ο παράγοντας p(C=c) υπολογίζεται βάσει της συχνότητας εμφάνισης της κλάσης c στα στιγμιότυπα του σώματος εκπαίδευσης. Οι δεσμευμένες πιθανότητες p(Xi = xi | Ci = c) υπολογίζονται ανάλογα με το αν το χαρακτηριστικό X2 είναι διακριτό ή συνεχές. Για τα διακριτά χαρακτηριστικά των διανυσμάτων, εκείνα δηλαδή που παίρνουν διακριτές τιμές, η πιθανότητα αυτή είναι ένας πραγματικός αριθμός, μεταξύ 0 και 1, ο οποίος αντιπροσωπεύει την πιθανότητα το χαρακτηριστικό Xi να πάρει την τιμή i x δεδομένης της κλάσης c . Για τα συνεχή χαρακτηριστικά, θεωρείται ότι οι τιμές ακολουθούν μια πιθανοτική κατανομή (ξεχωριστή για κάθε χαρακτηριστικό), η οποία προσεγγίζεται από τα διανύσματα εκπαίδευσης. Η πιο συνηθισμένη θεώρηση είναι οι τιμές των χαρακτηριστικών να είναι κανονικά κατανεμημένες. Οπότε για συνεχή χαρακτηριστικά ισχύει:

$$p(X_i = x_i \mid C = c) = g(x_i; \mu_{i,c}, \sigma_{i,c})$$

όπου, g η συνάρτηση πυκνότητας πιθανότητας μια κανονικής (Gaussian) κατανομής.

Το μοντέλο που δημιουργήσαμε αντιπροσωπεύει μια από τις πιο σημαντικές παραμέτρους για την εκπαιδευτική αξιολόγηση. Έχει μικρό αριθμό μεταβλητών που θα αξιολογηθούν από τον φορέα της εκπαίδευσης. Για κάθε κλάση και για κάθε διαφορετικό διακριτό χαρακτηριστικό, είναι απαραίτητο να γίνει αξιολόγηση της πιθανότητας του χαρακτηριστικού ώστε να λάβει κάθε δυνατή τιμή από τις πιθανές τιμές αυτού του χαρακτηριστικού, λαμβάνοντας υπόψη τη δεδομένη κλάση. Επιπλέον, ο ταξινομητής είναι σε θέση να υπολογίσει τις πιθανότητες p(C = + | X= x) και p(C = - | X= x) για ένα άγνωστο δείγμα x και να τις αντιστοιχίσει σε μία από τις μεγαλύτερες πιθανότητες μεταξύ ό-λων δείγματα.

Ερώτημα 3

ΣΥΝΑΡΤΗΣΗ ΓΙΑ CROSS VALIDATION

Για το ερώτημα 3 καθώς και για το ερώτημα 4 έχω δημιουργήσει μία συνάρτηση η οποία δέχεται σαν ορίσματα το μοντέλο ταξινομητή που θα χρησιμοποιήσουμε , τα δεδομένα εισόδου και εξόδου καθώς και την τιμή του fold.
Η χρήση του cross validation δίνοντας του (5-fold) θα χωρίσει τα δεδομένα μας σε 5 υποσύνολα. Τα 4 εξ αυτών θα χρησιμοποιηθούν για την εκπαίδευση ενώ το τελευταίο για το validation test.Το μοντέλο θα εκπαιδεύεται σε κάθε fold με διαφορετικά δεδομένα εκπαίδευσης και διαφορετικά δεδομένα επικύρωσης. Αυτό θα πραγματοποιείται 5 φορές όπου στο τέλος του κάθε fold θα υπολογίζεται η μέση απόδοση του μοντέλου μας(accuracy και loss που επιτυγχάνεται) .Αυτό θα βοηθήσει τόσο εμάς να υπολογίζουμε την απόδοση του με νέα testing δεδομένα όσο και το ίδιο στο να μπορεί να αποκτά την ικανότητα γενίκευσης. Έτσι κάθε φορά στο τέλος της εκπαίδευσης θα γνωρίζουμε μέσω της γραφικής παράστασης αν αποκλίνει η μπορεί να γενικεύσει.
Επίσης έχω δημιουργήσει επιπρόσθετα μεταβλητές οι οποίες μας επιστρέφουν αποτελέσματα όπως το geometric mean score μέσω της βιβλιοθήκης imblearn καθώς επίσης και το confusion_matrix.
Τέλος υπολογίζουμε τις μετρικές sensitivity και specificity για το μοντέλο που χρησιμοποιούμε.

```python
#Function to perform 5 Folds Cross-Validation

kf = KFold(n_splits=5)


def cross_validation(model, _X, _y, _cv=kf):
    # _X array input values
    # _Y out labels
    #cv Determines the number of folds for cross-validation.

     _scoring = ['accuracy', 'precision', 'recall', 'f1']

     #Model Training and validation with repeating
     results = cross_validate(estimator=model, X=_X,y=_y,cv=kf,scoring=_scoring,return_train_score=True,verbose=1)
     #Elegxoume tin apododsi pou petuxenei me ta test dedomena pou eginan split apo to kfold
     y_pred = cross_val_predict(model, X_train_minmax,  Y_train, cv=5)
     #pairnoume to accuracy
     accuracy = accuracy_score(Y_train, y_pred)
     #Geometric mean score
     geom_mean_score=geometric_mean_score(Y_train, y_pred)


     #Fit model without repeating (test prediction with split_testdata)
     model.fit(X_train_minmax,Y_train)
     test_pred = model.predict(X_test_minmax)
     test_accuracy=accuracy_score(Y_test,test_pred)

     print("REPORT CLASSIFICATION FOR train data with repeating 5fold:",classification_report(Y_train,y_pred))
     #info

     #Costum geometric mean calculation:
     sensitivity = recall_score(Y_train , y_pred,average='macro')
     specificity = recall_score(np.logical_not(Y_train) , np.logical_not(y_pred) , average='macro')
     geom_costum_score=sensitivity*specificity

     return {"Training Accuracy scores": results['train_accuracy'],
             "Mean Training Accuracy": results['train_accuracy'].mean()*100,
             "Training Precision scores": results['train_precision'],
             "Mean Training Precision": results['train_precision'].mean(),
             "Training Recall scores": results['train_recall'],
```

Sensitivity = (True Positive)/(True Positive + False Negative)
Μας δείχνει το μέτρο του ποσοστού των πραγματικά θετικών περιπτώσεων που έχουν προβλεφθεί ως θετικές .Η υψηλότερη τιμή θα σημαίνει υψηλότερη τιμή του αληθινού και χαμηλότερη τιμή των ψευδών αποτελεσμάτων. Εμείς επιθυμούμε μοντέλα με υψηλή ευαισθησία.

Specificity = (True Negative)/(True Negative + False Positive)
Ορίζεται η αναλογία των πραγματικά αρνητικών τιμών οι οποίες προβλέφθηκαν αρνητικές.

Οι δύο αυτές μετρικές χρησιμοποιούνται για την παράσταση της καμπύλης Roc Curve ενώ η περιοχή κάτω από την καμπύλη roc (auc) χρησιμοποιείται για τον προσδιορισμό της απόδοσης του μοντέλου.

ΣΥΝΑΡΤΗΣΗ ΓΙΑ ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ

Έπειτα έχω δημιουργήσει μια συνάρτηση η οποία μας επιστρέφει το bar graph για κάθε fold με τα αποτελέσματα εκπαίδευσης και επικύρωσης αντίστοιχα ενώ επιστρέφω και το geometric mean score.

```
from sklearn.metrics import accuracy_score
#'Function to plot a grouped bar chart showing the training and validation results of the ML model in each fold after applying K-fold cross-validation.
def plot_result(x_label, y_label, plot_title, X_train, Y_train,geometric_data):
    #x_label name algorithm
    #y_label: str, Name of metric being visualized e.g 'Accuracy'


        # Set size of plot
        plt.figure(figsize=(12,6))
        labels = ["1st Fold", "2nd Fold", "3rd Fold", "4th Fold", "5th Fold"]
        X_axis = np.arange(len(labels))
        ax = plt.gca()
        plt.ylim(0.40000, 1)
        plt.bar(X_axis+0.0, X_train, 0.2, color='blue', label='Training')
        plt.bar(X_axis+0.2, Y_train, 0.2, color='red', label='Validation')
        plt.bar(X_axis+0.4, geometric_data, 0.2, color='black', label='Geometric')
        plt.title(plot_title, fontsize=30)
        plt.xticks(X_axis, labels)
        plt.xlabel(x_label, fontsize=14)
        plt.ylabel(y_label, fontsize=14)
        plt.legend()
        plt.grid(True)
        plt.show()
```

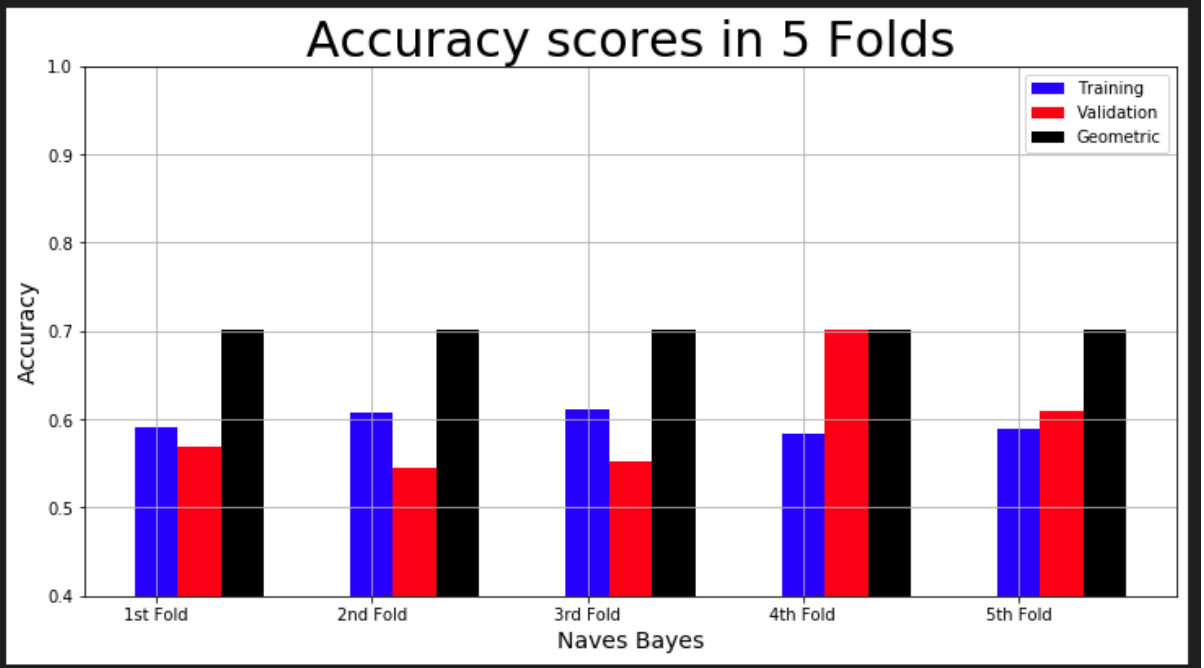
Ο ταξινομητής που χρησιμοποιήσαμε στο ερώτημα 3 είναι ο naive bayes με gaussian κατανομή. Έπειτα στην μεταβλητή results αποθηκεύουμε τα αποτελέσματα της συνάρτησης κλίσης.

```
#MODEL for Naive Bayes
gnb = GaussianNB()
kf = KFold(n_splits=5)

#decision_ result with callback functon
results= cross_validation(gnb, X_train_minmax, Y_train, kf)

#MEAN
Training_acc=results["Mean Training Accuracy"]
Validation_Acc=results["Mean Validation Accuracy"]

#PREDICTION ACCURACY WITH X_TEST && Y_TEST
pred_accuracy=results['Test_accuracy_score']
pred_accuracy
#Geometric Mean score
Geom =results['geometric_mean_score']
print(f'TRAINING ACCURACY :{Training_acc}'"\n",f'VALIDATION ACCURACY :{Validation_Acc}'"\n",f'PREDICTION ACCURACY WITH TEST DATA:{pred_accuracy}'"\n",F'GEOMETRIC_MEAN_SCORE:{Geom}'"


#prediction values
#print(results['Y_pred'])
```

ΑΠΟΤΕΛΕΣΜΑΤΑ

```
REPORT CLASSIFICATION FOR train data with repeating 5fold:

              1       0.94      0.48      0.64       316
              2       0.41      0.92      0.56       121

    accuracy                            0.60       437
   macro avg        0.67      0.70      0.60       437
weighted avg        0.79      0.60      0.62       437

TRAINING ACCURACY :59.61129758493655
 VALIDATION ACCURACY :59.5141065830721
 PREDICTION ACCURACY WITH TEST DATA:0.6095890410958904
 GEOMETRIC_MEAN_SCORE:0.6664552267830247
 GEOMETRIC_COSTUM_SCORE:0.70076629354535
```

```
#PLOT NAVE BAYES
model_name = "Naves Bayes"

plot_result(model_name,"Accuracy","Accuracy scores in 5 Folds",results["Training Acc

print(results['Y_pred'])
```




```
[2 1 1 2 2 2 2 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 1 2 1 2 2 2 2 2 2 2
 2 1 1 1 2 1 1 2 2 2 1 1 2 2 1 2 2 1 2 1 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2
 1 2 2 2 2 2 2 1 1 2 1 1 2 2 2 2 1 1 2 1 2 2 2 2 1 2 2 2 1 1 2 1 2 2 2 1 1
 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1 2 2 1 1 2 2 1 1 1 2 2 1 1 2 2 2 2 1 2 2 1 2 1
```

Παρατηρούμε ότι το μοντέλο μας πετυχαίνει ακρίβεια στην εκπαίδευση των δεδομένων στο 60% καθώς και παρόμοιο ποσοστό στο σύνολο επικύρωσης. Ετσι μπορούμε να συμπεράνουμε ότι το μοντέλο έχει την ικανότητα της γενίκευσης δηλαδή να μπορεί να προβλέπει νέα δεδομένα. Επίσης το geometric mean score είναι στο 70%.Ο γεωμετρικός μέσος όρος είναι καλύτερος από τον αριθμητικό μέσο όρο επειδή ο γεωμετρικός μέσος όρος είναι πιο ακριβής και αποτελεσματικός, όταν υπάρχει μεταβλητότητα στο σύνολο δεδομένων. Αυτός είναι ο λόγος που χρησιμοποιείται συχνά σε περιπτώσεις πρόγνωσης ασθένειας η στα χρηματοοικονομικά που μας ενδιαφέρει η ακρίβεια.