ΕΠΟΠΤΕΥΟΜΕΝΗ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΣΤΟΝ ΤΟΜΕΑ ΤΗΣ ΥΓΕΙΑΣ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ- ΒΑΣΙΚΕΣ ΈΝΝΟΙΕΣ ΚΑΙ ΕΦΑΡΜΟΓΕΣ

Χρυσάφης Θεόδωρος

Φεβρουάριος 2024

Πίνακας περιεχομένων

Ερώτηση....3
Απάντηση....4
Βασικές τεχνικές Μηχανικής Μάθησης....4
Παραλλαγές Νευρωνικών Δικτύων Βαθιάς Μάθησης....4
Προσφορά Τεχνητής Νοημοσύνης στον τομέα της υγείας....5
Παράδειγμα ανάλυσης βημάτων στο πλαίσιο των βασικών τεχνικών της Μηχανικής Μάθησης...6
Α. Συλλογή και προ-επεξεργασία δεδομένων....7
Β. Προετοιμασία Δεδομένων και Μηχανική Χαρακτηριστικών....8
Γ. Κατασκευή Μοντέλου....13
Δ. Αξιολόγηση Μοντέλου....15
Βιβλιογραφία....16

Ερώτηση

Να αναφερθούν επιγραμματικά οι βασικές τεχνικές Μηχανικής Μάθησης, οι παραλλαγές των νευρωνικών δικτύων που αφορούν τη Βαθιά Μάθηση, καθώς και πώς θα μπορούσε η Τεχνητή Νοημοσύνη να προσφέρει στον τομέα της υγείας. Δώστε ένα παράδειγμα και αναλύστε τα βήματα που πιστεύετε ότι ακολουθήθηκαν, ώστε να εκπαιδευτεί το αντίστοιχο μοντέλο του παραδείγματος που θα αναφέρετε.

Απάντηση

Βασικές τεχνικές Μηχανικής Μάθησης

Η Μηχανική Μάθηση μας προσφέρει εργαλεία τόσο για την ανάλυση πολύπλοκων δεδομένων όσο και για την ανακάλυψη καινούργιας γνώσης. Για την επίτευξη των παραπάνω ενεργειών χρησιμοποιούμε τεχνικές οι οποίες μας βοηθούν στην ανάλυση και επίλυση του προβλήματος που καλούμαστε να αντιμετωπίσουμε. Επιγραμματικά οι τεχνικές που χρησιμοποιούμε περιλαμβάνουν:

- την **Συλλογή των δεδομένων** (καθορισμός αξιόπιστης πηγής, έλεγχος ελλείψεων και ανακριβών δεδομένων),
- την **Προετοιμασία των δεδομένων** (καθαρισμός δεδομένων, κανονικοποίησης, κωδικοποίηση, επιλογή χαρακτηριστικών),
- την **Μηχανική χαρακτηριστικών** (επιλογή χαρακτηριστικών, δημιουργία χαρακτηριστικών, μετασχηματισμός χαρακτηριστικών με χρήση συνδυαστικής μετασχηματιστικής ή αλγοριθμικής εξέτασης στα αρχικά χαρακτηριστικά),
- την **Κατασκευή του μοντέλου** (επιλογή - εφαρμογή αλγόριθμου, εκπαίδευση σε ένα υποσύνολο των δεδομένων, επικύρωση σε ένα δεύτερο υποσύνολο δεδομένων για την αξιολόγηση της απόδοσης του μοντέλου),
- την **Αξιολόγηση του μοντέλου** (έλεγχος του μοντέλου σε σχέση με το πρόβλημα που χρήζει επίλυσης)

Παραλλαγές Νευρωνικών Δικτύων Βαθιάς Μάθησης

Η Βαθιά Μάθηση είναι μία υποκατηγορία της Μηχανικής Μάθησης η οποία αξιοποιεί τα Τεχνητά Νευρωνικά Δίκτυα (Νευρώνες, Σύνδεσμοι, Κρυμμένα Επίπεδα) για την ανάπτυξη συστημάτων που να μπορούν να μαθαίνουν από σύνθετα δεδομένα χωρίς την αξίωση κάποιας μετατροπής τους.

Έχουν αναπτυχθεί διάφορες αρχιτεκτονικές, αναλόγως του προβλήματος που καλούνται να επιλύσουν. Δύο από αυτές επιγραμματικά είναι:

- τα **Συνελικτικά Νευρωνικά Δίκτυα** - **Convolutional** N**eural Networks (CNN)** τα οποία σχεδιάστηκαν για την επεξεργασία δεδομένων τοπολογίας πλέγματος π.χ. εικόνων, βίντεο. Τμήματα των Συνελικτικών Νευρωνικών Δικτύων είναι:
  - τα **Συνελικτικά Επίπεδα** όπου με την χρήση συνελικτικών φίλτρων μετατρέπουν την εικόνα σε αριθμητικές τιμές, δημιουργώντας τους χάρτες χαρακτηριστικών,
  - τα **Pooling Επίπεδα (Επίπεδα Συγκέντρωσης)** όπου με την χρήση τους μειώνονται οι διαστάσεις των χαρτών χαρακτηριστικών
  - τα **Πλήρως Διασυνδεμένα Επίπεδα** στα οποία γίνεται η ανάλυση των χαρακτηριστικών και παράγεται η τελική πρόβλεψη
- τα **Επαναλαμβανόμενα Νευρωνικά Δίκτυα** - **Recurrent Neural Networks (RNNs)** τα οποία σχεδιάστηκαν για την επεξεργασία ακολουθιών δεδομένων π.χ. ανάλυση κειμένου, αναγνώριση ομιλίας, με κύριο χαρακτηριστικό την “μνήμη”, μιας εσωτερικής κατάστασης όπου επιτρέπει να διατηρούν πληροφορίες σχετικά με προηγούμενες εισόδους κατά την επεξεργασία νέων πληροφοριών.

  Το μέγεθος όμως της “μνήμης”, το πόσο πίσω μπορεί να “θυμηθεί” το RNN, είναι μία πρόκληση και γι’ αυτό τον λόγο αναπτύχθηκαν τα **Long Short-Term Memory (LSTM)** και **Gated Recurrent Units (GRU)** τα οποία διαχειρίζονται πολύ μεγαλύτερες ακολουθίες.

Προσφορά Τεχνητής Νοημοσύνης στον τομέα της υγείας

Η υιοθέτηση της Τεχνητής Νοημοσύνης στον τομέα της υγείας θα αναβαθμίσει ποιοτικά διαδικασίες τόσο στην διάγνωση και θεραπεία όσο και στην πρόληψη των ασθενειών. Η χρήση των **Συστημάτων Υποστήριξης Κλινικών Αποφάσεων - Clinical Decision Support System (CDSS)** μπορούν να υποστηρίξουν κλινικές αποφάσεις όταν ενσωματωθούν με ηλεκτρονικούς φακέλους ασθενών για ακριβέστερη και εξατομικευμένη διάγνωση, θεραπεία ή ενημέρωση.

Οι τομείς όπου τα CDSS έχουν την δυνατότητα να προσφέρουν βοήθεια είναι:

- στην **Ιατρική Απεικόνιση**, όπου με την ανάπτυξη αλγορίθμων Μηχανικής Μάθησης δίνεται η δυνατότητα στην ακριβέστερη και ταχύτερη αναγνώριση επιβλαβών χαρακτηριστικών σε ιατρικές εικόνες,
- στην **Προγνωστική Αναλυτική**, όπου με την χρήση στατιστικών αλγορίθμων, Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης δύναται η εξατομικευμένη πρόβλεψη της εξέλιξης ασθενειών, του εξατομικευμένου ρίσκου εμφάνισης ασθενειών, όπως και εξατομικευμένες θεραπείες βάση των ιστορικών δεδομένων των ασθενών.

Παράδειγμα ανάλυσης βημάτων στο πλαίσιο των βασικών τεχνικών της Μηχανικής Μάθησης

Το παράδειγμα στο οποίο θα αναλύσουμε τα βήματα που ακολουθήθηκαν στο πλαίσιο των βασικών τεχνικών της Μηχανικής Μάθησης αφορά μία ταξινόμηση συνοπτικών αλληλουχιών DNA. Ο τύπος της Μηχανικής Μάθησης είναι Επιβλεπόμενος (Supervised) επειδή η εκπαίδευση του μοντέλου γίνεται με την εισαγωγή δεδομένων εκπαίδευσης και των αντίστοιχων επιθυμητών αποτελεσμάτων. Με αυτήν την μέθοδο το μοντέλο μαθαίνει να ταξινομεί διάφορα δεδομένα στις επιθυμητές τιμές εξόδου.

Πριν ξεκινήσουμε, να σημειώσουμε ότι το περιβάλλον στο οποίο θα αναπτύξουμε και θα εκτελέσουμε το μοντέλο, είναι το Anaconda Jupyter. Επίσης σε κάθε στιγμιότυπο οθόνης (screenshot) που θα εμφανίζουμε στην παρούσα ανάλυση, θα περιέχει και σχόλια τα οποία θα επεξηγούν τον κώδικα.

Στην αρχή φορτώνουμε τις απαραίτητες βιβλιοθήκες (Εικόνα 1) που χρησιμοποιεί η ανάπτυξη του μοντέλου μας. Για λόγους σαφήνειας θα φορτώσουμε όλες τις βιβλιοθήκες στην αρχή. Θα μπορούσαμε να τις φορτώσουμε και κατά την διάρκεια ανάπτυξης, όποτε μας ήταν αναγκαίες για λόγους απόδοσης αλλά στην περίπτωση μας δεν συντρέχει κάποιος λόγος μιας και το σύνολο δεδομένων που θα χρησιμοποιήσουμε ως είσοδο είναι μικρό.

```
# Συναρτήσεις που αλληλεπιδρούν με τον διερμηνέα της python.
import sys
# Διαχείρηση πινάκων
import numpy as np
# Διαχείρηση συνόλου δεδομένων
import pandas as pd
# Εργαλεία επιλογής και αξιολόγησης μοντέλων
from sklearn import model_selection
# Φόρτωση ταξινομητών
from sklearn.neighbors import KNeighborsClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
# Λειτουργίες για τη μέτρηση της απόδοσης των μοντέλων
from sklearn.metrics import classification_report, accuracy_score
```

Εικόνα 1. Φόρτωση βιβλιοθηκών.

Στην συνέχεια δηλώνουμε μία συνάρτηση (Εικόνα 2) για εμφάνιση διάφορων πληροφοριών σχετικά με το υπολογιστικό σύστημα που χρησιμοποιούμε (Εικόνα 3).

```
'''
Μία απλή συνάρτηση για εμφάνιση πληροφοριών σχετικά με το
υπολογιστικό σύστημα ανάπτυξης του μοντέλου μας.
'''
def system_info():
    print("Pyhton version:\n", sys.version, "\n")
    print("Development System:\n", sys.platform.capitalize(), "\n")
    print("Conda Version:")
    # SList data type
    conda_version = !conda --version
    # cast to string
    print(' '.join(conda_version).capitalize())
```

Εικόνα 2. Δήλωση συνάρτησης system_info().

```
# Κλίση της συνάρτησης
system_info()
```

```
Pyhton version:
 3.9.18 (main, Sep 11 2023, 13:41:44)
[GCC 11.2.0]

Development System:
 Linux

Conda Version:
Conda 4.12.0
```

Εικόνα 3. Κλίση της συνάρτησης system_info() και το αποτέλεσμα της.

Α. Συλλογή και προ-επεξεργασία δεδομένων

Το πρώτο βήμα είναι να ορίσουμε την πηγή των δεδομένων που θα χρησιμοποιήσουμε. Στο παράδειγμα μας η πηγή των δεδομένων μας, προέρχεται από μία διεύθυνση URL πανεπιστημίου στο Internet. Τα δεδομένα τα εκχωρούμε σε μία μεταβλητή για να έχουμε την δυνατότητα να τα ελέγξουμε για τυχόν ελλείψεις ή ανακρίβειες και στην συνέχεια να τα επεξεργαστούμε (Εικόνα 4).

```
# Πηγή δεδομένων
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/\
molecular-biology/promoter-gene-sequences/promoters.data"
# Ορισμός χαρακτηριστικών
names = ['Class', 'id', 'Sequence']
# Φόρτωση των δεδομένων σε ένα dataframe με την βοήθεια της βιβλιοθήκης Pandas
data = pd.read_csv(url, names=names)
# Εμφάνιση μέρους των δεδομένων στην αρχή και στο τέλος
data.head(3), data.tail(3)
```

```
(  Class    id                                           Sequence
 0     +   S10  \t\ttactagcaatacgcttgcgttcggtggttaagtatgtataat...
 1     +  AMPC  \t\ttgctatcctgacagttgtcacgctgattggtgtcgttacaat...
 2     +  AROH  \t\tgtactagagaactagtgcattagcttatttttttgttatcat...,
     Class    id                                           Sequence
 103     -  1226  \t\tcgcgactacgatgagatgcctgagtgcttccgttactggatt...
 104     -   794  \t\tctcgtcctcaatggcctctaaacgggtcttgaggggtttttt...
 105     -  1442  \t\ttaacattaataaataaggaggctctaatggcactcattagcc...)
```

Εικόνα 4. Πρώτη επαφή με τα δεδομένα μας.

Ας δούμε πιο διεξοδικά τα χαρακτηριστικά Class και Sequence Εικόνα 5 και Εικόνα 6.

```
# Εμφάνιση χαρακτηριστικού Class.
classes = data.loc[:, "Class"]
# Η εντολή print() μπορεί να παραλειφθεί.
classes
```

```
0      +
1      +
2      +
3      +
4      +
      ..
101    -
102    -
103    -
104    -
105    -
Name: Class, Length: 106, dtype: object
```

Εικόνα 5. Χαρακτηριστικό Class.

```
# Εμφάνιση χαρακτηριστικού Sequence και εισαγωγή του σε λίστα
sequences = list(data.loc[:, "Sequence"])
sequences
```

```
['\t\ttactagcaatacgcttgcgttcggtggttaagtatgtataatgcgcgggcttgtcgt',
 '\t\ttgctatcctgacagttgtcacgctgattggtgtcgttacaatctaacgcatcgccaa',
 '\t\tgtactagagaactagtgcattagcttatttttttgttatcatgctaaccacccggcg',
 '\taattgtgatgtgtatcgaagtgtgttgcggagtagatgttagaatactaacaaactc',
 '\ttcgataattaactattgacgaaaagctgaaaaccactagaatgcgcctccgtggtag',
 '\taggggcaaggaggatggaaagaggttgccgtataaagaaactagagtccgtttaggt',
 '\t\tcaggggtggaggatttaagccatctcctgatgacgcatagtcagcccatcatgaat',
 '\t\ttttctacaaaacacttgatactgtatgagcatacagtataattgcttcaacagaaca',
 '\t\tcgacttaatatactgcgacaggacgtccgttctgtgtaaatcgcaatgaaatggttt',
 '\ttttttaaatttcctcttgtcaggccggaataactccctataatgcgccaccactgaca',
```

Εικόνα 6. Χαρακτηριστικό Sequence.

Παρατηρούμε ότι το χαρακτηριστικό Class αποτελείτε από χαρακτήρες άρα θα χρειαστεί να προβούμε σε κωδικοποίηση δηλαδή την μετατροπή των δεδομένων σε αριθμητικές τιμές. Στη λίστα sequences έχουμε την αλληλουχία των νουκλεοτιδίων όπου a = Adenine, c = Cytosine, g = Guanine και t = Thymine και περιέχουν την γενετική πληροφορία στο DNA. Παρατηρούμε ότι έχουμε δύο χαρακτήρες “\t” οι οποίοι δεν αποτελούν μέρος της αλληλουχίας και θα πρέπει να τους εξαλείψουμε.

Β. Προετοιμασία Δεδομένων και Μηχανική Χαρακτηριστικών

Η Προετοιμασία των Δεδομένων και η Μηχανική Χαρακτηριστικών είναι τα επόμενα δύο βήματα τα οποία θα προχωρήσουμε σύμφωνα με τις παρατηρήσεις του προηγούμενου βήματος της Συλλογής και Προ-επεξεργασίας των δεδομένων Εικόνα 7.

```
# Δημιουργία λεξικού dataset {index: ["νουκλεοτιδιο_1",...,"νουκλεοτίδιο_n", "κλάση"]}
dataset = {}
for i, seq in enumerate(sequences):
    nucleotides = list(seq)
    nucleotides = [x for x in nucleotides if x != '\t']
    nucleotides.append(classes[i])
    dataset[i]=nucleotides

# Εμφάνιση τριών πρώτων και τριών τελευταίων εγγραφών της πρώτης γραμμής του λεξικού
dataset[0][:3], dataset[0][-3:]
```

```
(['t', 'a', 'c'], ['g', 't', '+'])
```

Εικόνα 7. Προετοιμασία δεδομένων αλληλουχίας νουκλεοτιδίων.

Στην συνέχεια θα δημιουργήσουμε ένα dataframe από το παραπάνω dataset και θα εμφανίσουμε τις διαστάσεις του Εικόνα 8.

```
# Μετατροπή του λεξικού σε dataframe
dframe = pd.DataFrame(dataset)
# Διαστάσεις του dataframe
dframe.shape
```

```
(58, 106)
```

Εικόνα 8. Διαστάσεις του νέου Dataframe.

Όπως βλέπουμε τις διαστάσεις του νέου dataframe, θα χρειαστεί να μετατρέψουμε τις στήλες σε γραμμές Εικόνα 9.

```
# Μετατόπιση γραμμών του dataframe σε στήλες
df = dframe.transpose()
df
```

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	57
0	t	a	c	t	a	g	c	a	a	t	...	g	c	t	t	g	t	c	g	t	+
1	t	g	c	t	a	t	c	c	t	g	...	c	a	t	c	g	c	c	a	a	+
2	g	t	a	c	t	a	g	a	g	a	...	c	a	c	c	c	g	g	c	g	+

Εικόνα 9. Το νέο dataframe με τα δεδομένα.

Στην συνέχεια το χαρακτηριστικό 57 θα το μετονομάσουμε σε “Class” Εικόνα 10.

```
# Μετονομασία του χαρακτηριστικού 57 σε λεκτικό "Class"
df.rename(columns={57:'Class'}, inplace=True)
df.head(3), df.tail(3)
```

```
(   0  1  2  3  4  5  6  7  8  9  ... 48 49 50 51 52 53 54 55 56 Class
 0  t  a  c  t  a  g  c  a  a  t  ...  g  c  t  t  g  t  c  g  t     +
 1  t  g  c  t  a  t  c  c  t  g  ...  c  a  t  c  g  c  c  a  a     +
 2  g  t  a  c  t  a  g  a  g  a  ...  c  a  c  c  c  g  g  c  g     +

 [3 rows x 58 columns],
      0  1  2  3  4  5  6  7  8  9  ... 48 49 50 51 52 53 54 55 56 Class
 103  c  g  c  g  a  c  t  a  c  g  ...  a  a  g  g  c  t  t  c  c     -
 104  c  t  c  g  t  c  c  t  c  a  ...  a  g  g  a  g  g  a  a  c     -
 105  t  a  a  c  a  t  t  a  a  t  ...  t  c  a  a  g  a  a  c  t     -

 [3 rows x 58 columns])
```

Εικόνα 10. Μετονομασία χαρακτηριστικού.

Μπορούμε επίσης να δούμε τον αριθμό των νουκλεοτιδίων του ιδίου τύπου που αποτελείται η κάθε αλληλουχία και να δημιουργήσουμε ένα νέο dataframe Εικόνα 11 και Εικόνα 12.

```
# Δημιουργία λίστας με την ποσότητα των νουκλεοτιδίων ανά νουκλεοτίδιο
# που περιέχονται στις αλληλουχίες
series = []
for name in df.columns:
    series.append(df[name].value_counts())

series[:1], series[-1:]
```

```
([t    38
  c    27
  a    26
  g    15
  Name: 0, dtype: int64],
 [+    53
  -    53
  Name: Class, dtype: int64])
```

Εικόνα 11. Εμφάνιση αριθμού νουκλεοτιδίων του ιδίου τύπου στην αλληλουχία.

```
# Νέο dataframe με τον αριθμό των νουκλεοτιδίων ανά τύπο
info = pd.DataFrame(series)
info
```

	t	c	a	g	+	-
0	38.0	27.0	26.0	15.0	NaN	NaN
1	26.0	22.0	34.0	24.0	NaN	NaN
2	27.0	21.0	30.0	28.0	NaN	NaN
3	26.0	30.0	22.0	28.0	NaN	NaN

Εικόνα 12. Νέο dataframe με τον αριθμό των νουκλεοτιδίων ανά τύπο.

Για καλύτερη ανάγνωση θα μετατρέψουμε τις στήλες σε γραμμές του νέου dataframe Εικόνα 13.

```
# Μετατροπή γραμμών σε στήλες.
details = info.transpose()
details
```

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	Class
t	38.0	26.0	27.0	26.0	22.0	24.0	30.0	32.0	32.0	28.0	...	21.0	22.0	23.0	33.0	35.0	30.0	23.0	29.0	34.0	NaN
c	27.0	22.0	21.0	30.0	19.0	18.0	21.0	20.0	22.0	22.0	...	36.0	42.0	31.0	32.0	21.0	32.0	29.0	29.0	17.0	NaN
a	26.0	34.0	30.0	22.0	36.0	42.0	38.0	34.0	33.0	36.0	...	23.0	24.0	28.0	27.0	25.0	22.0	26.0	24.0	27.0	NaN
g	15.0	24.0	28.0	28.0	29.0	22.0	17.0	20.0	19.0	20.0	...	26.0	18.0	24.0	14.0	25.0	22.0	28.0	24.0	28.0	NaN
+	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	53.0
-	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	53.0

6 rows × 58 columns

Εικόνα 13. Μετατροπή στηλών σε γραμμές.

Παρατηρούμε ότι οι γραμμές αυτές περιέχουν πληροφορία κατηγοριών δηλαδή αν η αλληλουχία των νουκλεοτιδίων ανήκει στην κατηγορία “t” ή την κατηγορία “+” κ.λ.π.. Για την διαχείριση τους χρησιμοποιούμε την βιβλιοθήκη Pandas και την συνάρτηση get_dummies() Εικόνα 14.

```
# Μετατροπή κατηγορηματικών τιμών με dummy τιμές (0, 1)
numerical_df = pd.get_dummies(df)
numerical_df
```

	0_a	0_c	0_g	0_t	1_a	1_c	1_g	1_t	2_a	2_c	...	55_a	55_c	55_g	55_t	56_a	56_c	56_g	56_t	Class_+	Class_-
0	0	0	0	1	1	0	0	0	0	1	...	0	0	1	0	0	0	0	1	1	0
1	0	0	0	1	0	0	1	0	0	1	...	1	0	0	0	1	0	0	0	1	0
2	0	0	1	0	0	0	0	1	1	0	...	0	1	0	0	0	0	1	0	1	0

Εικόνα 14. Κατηγοριοποίηση νουκλεοτιδίων ανά εμφάνιση στην αλληλουχία.

Συνέχεια της Μηχανικής Χαρακτηριστικών θα διαγράψουμε το πλεονάζων χαρακτηριστικό “Class_-” επειδή περιέχει πληροφορία που μπορεί να ανακτηθεί και από το χαρακτηριστικό “Class_+”, θα μετονομάσουμε το χαρακτηριστικό “Class_+” σε “Class” και θα αποθηκεύσουμε το αποτέλεσμα σε ένα νέο dataframe Εικόνα 15.

```
# Διαγραφή πλεονάζων χαρακτηριστικού "Class_-"
df = numerical_df.drop(columns=['Class_-'])
# Μετονομασία χαρακτηριστικού "Class_+" σε "Class"
df.rename(columns={'Class_+': 'Class'}, inplace=True)
df
```

	0_a	0_c	0_g	0_t	1_a	1_c	1_g	1_t	2_a	2_c	...	54_t	55_a	55_c	55_g	55_t	56_a	56_c	56_g	56_t	Class
0	0	0	0	1	1	0	0	0	0	1	...	0	0	0	1	0	0	0	0	1	1
1	0	0	0	1	0	0	1	0	0	1	...	0	1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	0	1	1	0	...	0	0	1	0	0	0	0	1	0	1

Εικόνα 15. Μηχανική Χαρακτηριστικού Class.

Στην συνέχεια θα δημιουργήσουμε τα υποσύνολα εκπαίδευσης και ελέγχου των δεδομένων μας Εικόνα 16, Εικόνα 17.

```
# Δημιουργία υποσυνόλων δεδομένων για την εκπαίδευση του μοντέλου.
X = np.array(df.drop(['Class'],axis=1))
X
```

```
array([[0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 1, 0],
       ...,
       [0, 1, 0, ..., 1, 0, 0],
       [0, 1, 0, ..., 1, 0, 0],
       [0, 0, 0, ..., 0, 0, 1]], dtype=uint8)
```

Εικόνα 16. Υποσύνολο δεδομένων για εκπαίδευση του μοντέλου.

```
# Δημιουργία υποσυνόλων δεδομένων για τον έλεγχο του μοντέλου.
y = np.array(df['Class'])
y
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=uint8)
```

Εικόνα 17. Υποσύνολο δεδομένων για έλεγχο του μοντέλου.

Τα δύο παραπάνω υποσύνολα δεδομένων θα τα χωρίσουμε σε ακόμη δύο διαφορετικά υποσύνολα δεδομένων για να δοκιμάσουμε την απόδοση του μοντέλου μας Εικόνα 18, Εικόνα 19 και Εικόνα 20.

```
# Δημιουργία τυχαίων υποσυνόλων δεδομένων για την εκπαίδευση και τον έλεγχο του μοντέλου.
# Καταχώρηση σταθερής τιμής seed για να μας επιστρέφει πάντα το ίδιο αποτέλεσμα.
seed = 1
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.25,
                                                                    random_state=seed)
```

Εικόνα 18. Δημιουργία υποσυνόλων δεδομένων εκπαίδευσης και ελέγχου.

```
X_train, y_train
```

```
(array([[0, 0, 0, ..., 0, 1, 0],
        [0, 0, 0, ..., 0, 1, 0],
        [0, 0, 0, ..., 0, 0, 1],
        ...,
        [0, 0, 0, ..., 1, 0, 0],
        [0, 0, 1, ..., 0, 0, 0],
        [0, 1, 0, ..., 0, 0, 1]], dtype=uint8),
 array([0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
        0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
        1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1], dtype=uint8))
```

Εικόνα 19. Εμφάνιση υποσυνόλων εκπαίδευσης.

```
X_test, y_test
```

```
(array([[0, 0, 0, ..., 0, 0, 1],
        [0, 1, 0, ..., 1, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 1, 0, ..., 0, 0, 1],
        [0, 0, 0, ..., 0, 1, 0],
        [0, 0, 0, ..., 0, 0, 1]], dtype=uint8),
 array([0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
        0, 1, 1, 1, 0], dtype=uint8))
```

Εικόνα 20. Εμφάνιση υποσυνόλων ελέγχου.

Γ. Κατασκευή Μοντέλου

Σε συνέχεια της ανάλυσης μας θα προχωρήσουμε στην κατασκευή του μοντέλου που θα εκπαιδεύσουμε. Επειδή θα εκπαιδεύσουμε το μοντέλο με επτά ταξινομητές θα τους καταχωρήσουμε σε λίστα Εικόνα 21 ώστε με μία επανάληψη να έχουμε τα επτά αποτελέσματα.

```
# Δήλωση των ταξινομητών σε λίστα και της μετρικής "accuracy"
# για την αξιολόγηση του μοντέλου.
scoring = 'accuracy'
names = ['Nearest Neighbors', 'Gaussian Process', 'Decision Tree', 'Random Forest',
         'AdaBoost', 'Naive Bayes', 'SVM Linear']
classifiers = [KNeighborsClassifier(n_neighbors=3),
               GaussianProcessClassifier(kernel=None), # None = ("1.0, * RBF(1.0)")
               DecisionTreeClassifier(max_depth=5),
               RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
               AdaBoostClassifier(), GaussianNB(), SVC(kernel='linear'),]
```

Εικόνα 21. Δήλωση ταξινομητών σε λίστα.

Στην συνέχεια δημιουργούμε ένα tuple models(“περιγραφή ταξινομητη”, ταξινομιτής) Εικόνα 22.

```
# Δημιουργία tuple models(name, classifier)
models = zip(names, classifiers)
for model in models:
    print(model)
```

```
('Nearest Neighbors', KNeighborsClassifier(n_neighbors=3))
('Gaussian Process', GaussianProcessClassifier())
('Decision Tree', DecisionTreeClassifier(max_depth=5))
('Random Forest', RandomForestClassifier(max_depth=5, max_features=1, n_estimators=10))
('AdaBoost', AdaBoostClassifier())
('Naive Bayes', GaussianNB())
('SVM Linear', SVC(kernel='linear'))
```

Εικόνα 22. Δημιουργία tuple περιγραφής ταξινομιτών και ταξινομιτών με τις παραμέτρους τους.

Τέλος, σε μία δομή επανάληψης εκπαιδεύουμε τα μοντέλα, εκτελούμε πρόβλεψη Εικόνα 23 και εμφανίζουμε τα αποτελέσματα των μετρικών για καθέναν από τους επτά ταξινομιτές Εικόνα 24. Μετά την εμφάνιση των αποτελεσμάτων καλούμαστε να αξιολογήσουμε τα μοντέλα και τα αποτελέσματα βάση των εμφανιζόμενων αναφορών και μετρικών.

```
# Εκπαίδευση μοντέλου, πρόβλεψη και εμφάνιση αποτελεμάτων
# για την αξιολόγηση της απόδοσης του
results = []
names_ = []

for name, model in models:
    # Δημιουργία επικυρωτή με 10 "καλάθια" δεδομένων
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names_.append(name)
    # Εμφάνιση μετρικών αποτελεσμάτων
    msg = "%s: mean:%f, std:%f" % (name, cv_results.mean(), cv_results.std())
    print(msg)
    # Εκπαίδευση μοντέλου με δεδομένα εκπαίδευσης
    model.fit(X_train, y_train)
    # Πρόβλεψη με δεδομένα ελέγχου
    predictions = model.predict(X_test)
    # Εμφάνιση μετρικής "Accuracy" στο αποτέλεσμα της πρόβλεψης
    print(name,'Accuracy: ', accuracy_score(y_test, predictions))
    # Εμφάνιση
    print(name,'Report:')
    print(classification_report(y_test, predictions))
    print('-' * 55, '\n')
```

Εικόνα 23. Εκπαίδευση, επικύρωση και εμφάνιση αποτελεσμάτων μοντέλου.

```
Nearest Neighbors: mean:0.837500, std:0.125623
Nearest Neighbors Accuracy:  0.7777777777777778
Nearest Neighbors Report:
              precision    recall  f1-score   support

           0       1.00      0.65      0.79        17
           1       0.62      1.00      0.77        10

    accuracy                           0.78        27
   macro avg       0.81      0.82      0.78        27
weighted avg       0.86      0.78      0.78        27

------------------------------------------------------
```

```
Gaussian Process: mean:0.785714, std:0.136464
Gaussian Process Accuracy:  0.8148148148148148
Gaussian Process Report:
              precision    recall  f1-score   support

           0       1.00      0.71      0.83        17
           1       0.67      1.00      0.80        10

    accuracy                           0.81        27
   macro avg       0.83      0.85      0.81        27
weighted avg       0.88      0.81      0.82        27

------------------------------------------------------
```

```
Decision Tree: mean:0.735714, std:0.151059
Decision Tree Accuracy:  0.8518518518518519
Decision Tree Report:
              precision    recall  f1-score   support

           0       1.00      0.76      0.87        17
           1       0.71      1.00      0.83        10

    accuracy                           0.85        27
   macro avg       0.86      0.88      0.85        27
weighted avg       0.89      0.85      0.85        27

------------------------------------------------------
```

```
Random Forest: mean:0.671429, std:0.185955
Random Forest Accuracy:  0.5925925925925926
Random Forest Report:
              precision    recall  f1-score   support

           0       0.75      0.53      0.62        17
           1       0.47      0.70      0.56        10

    accuracy                           0.59        27
   macro avg       0.61      0.61      0.59        27
weighted avg       0.65      0.59      0.60        27

------------------------------------------------------
```

```
Naive Bayes: mean:0.837500, std:0.137500
Naive Bayes Accuracy:  0.9259259259259259
Naive Bayes Report:
              precision    recall  f1-score   support

           0       1.00      0.88      0.94        17
           1       0.83      1.00      0.91        10

    accuracy                           0.93        27
   macro avg       0.92      0.94      0.92        27
weighted avg       0.94      0.93      0.93        27

------------------------------------------------------
```

```
AdaBoost: mean:0.925000, std:0.114564
AdaBoost Accuracy:  0.8518518518518519
AdaBoost Report:
              precision    recall  f1-score   support

           0       1.00      0.76      0.87        17
           1       0.71      1.00      0.83        10

    accuracy                           0.85        27
   macro avg       0.86      0.88      0.85        27
weighted avg       0.89      0.85      0.85        27

------------------------------------------------------
```

```
SVM Linear: mean:0.850000, std:0.108972
SVM Linear Accuracy:  0.9629629629629629
SVM Linear Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        17
           1       0.91      1.00      0.95        10

    accuracy                           0.96        27
   macro avg       0.95      0.97      0.96        27
weighted avg       0.97      0.96      0.96        27

------------------------------------------------------
```

Εικόνα 24. Εμφάνιση αποτελεσμάτων εκπαίδευσης.

Δ. Αξιολόγηση Μοντέλου

Η αξιολόγηση του μοντέλου είναι μία κρίσιμη διαδικασία διότι πέρα από την σωστή ερμηνεία των αποτελεσμάτων θα πρέπει να προσέξουμε να μην παρουσιαστεί το φαινόμενο της υπερεκπαίδευσης. Η προετοιμασία, η μηχανική χαρακτηριστικών, η κατασκευή και αξιολόγηση του μοντέλου είναι μία επαναλαμβανόμενη διαδικασία για τον λόγο ότι πιθανών να μην έχουμε σωστή ανταπόκριση του μοντέλου στην απάντηση που καλούμαστε να δώσουμε εξαρχής. Επιπλέον θα μπορούσαμε να οπτικοποιήσουμε με γραφήματα τα αποτελέσματα των ταξινομιτών για περαιτέρω ανάλυση και παρουσίαση των αποτελεσμάτων.

Στην παρούσα αξιολόγηση των αποτελεσμάτων και συγκρίνοντας την μετρική Accuracy (Ορθότητα) διαπιστώνουμε ότι ο ταξινομιτής SVM Linear έχει καλύτερη απόδοση (0,96) από τους υπόλοιπους το οποίο σημαίνει ότι ο ταξινομητής ανταποκρίνεται σωστά σε νέα δεδομένα κατά 96%.

Βιβλιογραφία

Αντωνόπουλος Χρήστος, “Εποπτευόμενη Μηχανική Μάθηση στον Τομέα της Υγείας”.

Εικόνα Εξόφυλλου, https://www.healthweb.gr/perissotera/texnologia/ai-ygeia-oi-perissoteroi-astheneis-exoun-thetikes-apopseis-gia-tin-texniti-noimosyni-stin-iatriki.

Νουκλεοτίδιο, https://el.wikipedia.org/wiki/%CE%9D%CE%BF%CF%85%CE%BA%CE%BB%CE%B5%CE%BF%CF%84%CE%AF%CE%B4%CE%B9%CE%BF.

AdaBoostClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn-ensemble-adaboostclassifier.

Convolutional neural network, https://en.wikipedia.org/wiki/Convolutional_neural_network.

DecisionTreeClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn-tree-decisiontreeclassifier.

GaussianNB, https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html.

GaussianProcessClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessClassifier.html#sklearn-gaussian-process-gaussianprocessclassifier.

KneighborsClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KneighborsClassifier.html#sklearn-neighbors-kneighborsclassifier.

RandomForestClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn-ensemble-randomforestclassifier.

Recurrent neural network, https://en.wikipedia.org/wiki/Recurrent_neural_network.

SVC, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.