RAPORTARE STIINTIFICA

IONIS – NITICSplus - solutii pentru prevenirea dementei

Etapa 3: Implementarea si integrarea sistemului

(1.01.2019 – 31.12.2019)

Contract: Nr. 53/2017 / AAL2017-AAL-2016-074-IONIS-2 / IONIS

Contents

1. Scopul proiectului

Persoanele cu dementa demonstreaza, inca din stadiul incipient al bolii, pierderea memoriei, episoade de confuzie si dezorientarea in spatiu si timp. Acestea din urma se intampla adesea in mod imprevizibil, cu consecinte care uneori pot fi foarte grave. Odata cu evolutia bolii, pierderea si ratacirea pot deveni evenimente simptomatice si recurente. In prezent, membrii familiilor care, fara alte solutii, decurg

adesea la masuri drastice, reducand libertatea pacientului. Daca acest lucru nu este posibil, persoanele care au grija de un pacient cu dementa traiesc intr-o stare de stres continuu. Serviciile bazate pe localizare au potentialul de a oferi, cu un cost accesibil, un ajutor util pentru persoanele care ingrijesc persoanele cu dementa. Motivat de acest context, propunem dezvoltarea proiectului IONIS prin exploatarea serviciilor bazate pe localizare, monitorizarea calitatii somnului, servicii de comunicare pentru a oferi o gama larga de solutii specifice dementei prin sprijinul continuu pentru persoanele cu dementa atunci cand sunt acasa sau in exterior. Pentru a dezvolta si a valida solutia IONIS, se va utiliza un proiect centrat pe utilizator care implica incercari ample de testare.
In cadrul proiectului – etapa 3, UPB a realizat implementarea recunoasterii activitatilor umane in vederea eliminarii detectiilor false in identificarea rataciriilor in interiorul casei, metodei pentru detectia rataciriilor (in interior cat si in exteriorul casei), implementarea metodei de invatare a profilului utilizatorului bazat pe datele preluate din timpul somnului, precum si la realizarea interfetei folosind comenzi vocale.

2. Obiective etapa 2019

Obiectivele etapei 2019 au constat in implementarea activitatilor din planul de realizare: implementarea, testarea si optimizarea componentelor aferente UPB. In acest context a fost extinsa metoda de identificare a rataciriilor persoanelor supravegheate in interiorul casei prin utilizarea recunoasterea activitatilor (pentru a putea elimina din detectiile false). Totodata a fost extinsa metoda de corelare a parametrilor achizitionati in tmpul somnului in vederea invatarii modelelor adaptate fiecarui utilizator, respectiv fiecarui utilizator in vederea identificarii unor posibile probleme. Activitatile prevazute pentru 2019 sunt urmatoarele:

- Activitatea 3.1: Contributii la dezvoltarea modulelor sistemului
- Activitatea 3.2: Contributii la realizarea sistemului integrat
- Activitatea 3.3: Analiza datelor colectate in experimente realizate pe teren
- Activitatea 3.4: Contributii la optimizarea platformei integrate
- Activitatea 3.5: Participare la manifestari stiintifice, vizite de lucru

3. Rezumatul etapei

Obiectivele au fost realizate integral, gradul de atingere al rezultatelor fiind de 100%. **Activitatea 3.1** a avut ca scop extinderea implementarilor realizate in etapa 2 a modulelor de detectie a rataciriilor (in interiorul casei), precum si extragerea de informatii referitoare la corelarea parametrilor corespunzatori tulburarilor de somn. In cadrul **activitatii 3.2** UPB a contribuit la dezvoltarea interfetei sistemului, utilizand interactiunea intre utilizator si sistem prin comenzi vocale. **Activitatea 3.3** a avut drept scop testarea modulelor implementate in cadrul activitatii 3.1. In cadrul **activitatii 3.4**, optimizarile realizate au avut in vedere fine tuning pentru modelele referitoare la reconasterea activitatilor, modelele invatate pentru valorile referitoare la parametri de somn cat si pentru comenzile vocale. **Activitatea 3.5** a avut ca scop participarea la manifestari stiintifice, precum si vizite de lucru – in cadrul acestei etape a fost prezentat posterul ce descrie aspectele principale ale proiectului in cadrul forumului AAL (23-25 Septembrie 2019), Aarhus, Danemarca, au fost realizate 4 articole stiintifice: 1 articol in curs de evaluare trimis la un jurnal clasificat Q1, 1 lucrare acceptata la conferinta cu proceedings Springer si 2 lucrari in curs de evaluare la conferinta cu proceedings indexat ISI. De asemenea au avut loc 2 intalniri cu membrii consortiului (17-18 iunie 2019, Varsovia, Polonia, respectiv 9-10 noiembrie 2019, Nova Gorica, Slovenia) in care s-a discutat stadiul curent al proiectului, si s-a realizat planificarea activitatilor urmatoare in cadrul proiectului.

4. Descriere activitati

Activitatea 3.1: Contributii la dezvoltarea modulelor sistemului in cadrul acestei activitati a fost extinsa metoda de detectie a ratacilor prin utilizarea recunoasterii activitatilor zilnice in vederea eliminarii detectiilor false din identificarea ratacirilor in interior, precum si extinderea monitorizarii calitatii somnului, prin corelarea diferitilor parametri masurati in timpul somnului.

Recunoasterea activitatilor zilnice - Prima versiune pentru detectia ratacirilor in interior cat si monitorizarea calitatii somnului au fost realizate si descrise in etapa 2 (2018) aferenta proiectului. Ratacirea este unul dintre comportamentele cele mai frecvente, problematice si daunatoare pentru persoanele cu dementa, care este strans asociata cu evenimente adverse cum ar fi caderea sau pierderea. Pe baza studiilor medicale, ratacirea poate fi definita ca:

(1) miscarea inainte si inapoi intre oricare doua puncte;
(2) miscarea circulara reluand unele puncte secvential de-a lungul unei cai;
(3) miscare aleatoare, fara repetarea punctelor intr-o secventa de deplasare.

Recunoasterea activitatilor zilnice a fost realizata prin dezvoltarea unei aplicatii bazate pe retele neurale, fiind formata din doua module, o retea convolutionala si o retea recurenta. Pentru implementarea primului modul, a fost utilizata o arhitectura ce foloseste convolutii in doua dimensiuni, dar si una care foloseste convolutii in trei dimensiuni. Se doreste folosirea caracteristicilor spatiale, dar si observarea evolutiei in timp a a unui clip video, lucru obtinut folosind retele recurente de tip *Long Short Term Memory* (LSTM). Folosind informatia invatata in primul modul de catre retelele convolutionale, a doua parte imbunatateste intelegerea cadrelor de catre intreaga retea. Prin evaluarea celor doua metode, rezultatele obtinute relativ apropiate de ceea ce s-a raportat in *state-of-the-art.*

Proiectare reta bazata pe o retea convolutionala 2D si o retea recurenta: Acest model presupune extragerea caracteristicilor spatiale din cadrele primite la intrare, folosind o retea convolutionala in doua dimensiuni si folosirea informatiei invatate, mai departe, in partea a doua, intr-o retea recurenta. Prin folosirea datelor din dimensiunea temporala si a recurentelor, se va mima memoria umana ce ia decizii pe baza perceptiei de la momentul curent dar si pe baza informatiilor deja cunoscute de la momentele de timp anterioare. Astfel, in plus fata de o reteaua convolutionala clasica, se va capta informatia temporala ce va folosi la diferentierea si mai buna intre clase. Reteaua convolutionala foloseste la intrare cadre RGB, iar reteaua recurenta primeste la fiecare moment de timp un vector de caracteristici obtinut de reteaua convolutionala. Intreaga arhitectura a modelului se poate observa in Figura 1.
Reteaua convolutionala primeste la intrare cadre de tip RGB de dimensiune 224x224. Primul strat de convolutie, implementat in Keras folosind obiectul de tip Layer, Conv2D, foloseste 96 de filtre de dimensiune 7x7 si astfel produce 96 de harti de activare de dimensiune 112x112. In continuare, se adauga un strat de esantionare si un altul de normalizare. Prin esantionare, dimensiunea hartilor de activare se reduce la 55x55. Se folosesc vecinatati de dimensiune 3x3 cu un pas de 2 din care se extrag valorile maxime ce se vor regasi in fiecare harta de activare obtinuta de stratul MaxPooling2D. Urmatorul strat, cel de BatchNormalization separa datele si aplica o operatie de normalizare asupra fiecarei grupari. Operatia de normalizare presupune scaderea mediei tuturor valorilor si impartirea la deviatia standard. Urmatoarele trei nivele sunt tot o insiruire formata din straturile Conv2D, MaxPooling2D, BatchNormalization, care folosesc aceleasi principii de functionare explicate mai sus. In urma aplicarii acestor operatii se produc 384 de harti de activare de dimensiune 13x13 deoarece stratul de convolutie foloseste 384 de filtre de dimensiune 5x5, iar filtrul folosit in stratul de pooling are dimensiune 3x3. Reteaua se continua cu inca 3 straturi de convolutie care vor aplica pe rand 512, 512, respectiv 384 de filtre de dimensiune 3x3. Hartile de activare obtinute isi vor reduce dimensiunea printr-un strat de MaxPooling ce foloseste filtre de 3x3 si un pas de 2. Astfel, dupa aplicarea acestui strat, reteaua contine 384 de harti de dimensiune 6x6.

Functia de activare folosita, LeakyRelu, este implementata in Keras sub forma unui strat denumit LeakyReLu care se adauga dupa straturile de convolutie, cele care au nevoie de activare. Parametrul alpha, ce determina panta negativa, este la valoarea 0.1.
Reteaua recurenta are nevoie sa primeasca un vector de caracteristici pentru fiecare moment de timp. Prin urmare, este nevoie de modificarea dimensiunii datelor din 2D in 1D, operatie implementata in Keras de stratul de tip Flatten care transforma cele 384 de harti de activare de dimensiune 6x6 si produce un vector unidimensional de 13824 de valori. Complet conectat la acest strat de aplatizare este un strat clasic de tip Dense ce contine 4096 de noduri. Acesta foloseste LeakyRelu ca functie de activare, de asemenea implementat printr-un obiect de tip Layer. Reteaua convolutionala se termina cu acest strat de Dropout si produce un vector unidimensional de 4096 de caracteristici. Reteaua recurenta contine un strat de neuroni de tip LSTM, in numar de 256. Atat intrarea cat si iesirea acestui strat sunt de tip multi, adica se primeste un vector de caracteristici si se returneaza un altul pentru fiecare cadru dintr-un video. Iesirea stratului retelei recurente va avea o dimensiune egala cu numarul de neuroni, pentru fiecare moment de timp. Clasificarea propriu-zisa se realizeaza cu un strat Dense complet conectat, cu 101 neuroni, numarul de clase existente in setul de date folosit, astfel se obtine o probabilitate pentru fiecare clasa. Scopul este de a clasifica un intreg video folosind rezultatele obtinute pentru fiecare cadru. In acest scop, s-a ales stratul GlobalAveragePooling1D pentru a calcula o medie intre valorile existente pe dimensiunea temporala. Acest strat produce un vector de 101 valori ce reprezinta predictia retelei.

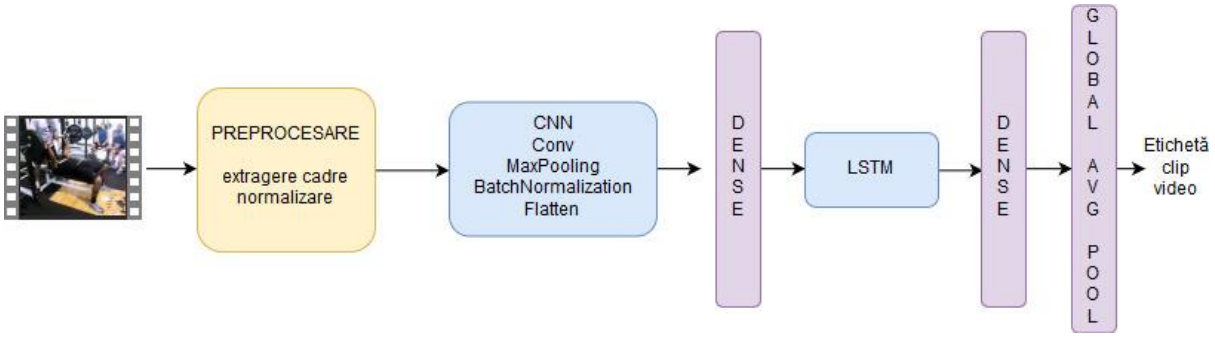


Figura 1. Arhitectura modelului format din convolutii 2D si recurente

Proiectarea modelului bazat pe o retea convolutionala 3D si o retea recurenta: Retelele convolutionale in doua dimensiuni pot fi usor extinse pentru a obtine la iesire date sub forma de volume, folosind filtre in trei dimensiuni, fiind potrivite pentru prelucrare si extragere de cunostinte spatio-temporale din video. Bazat pe principiul de a folosi retele convolutionale si retele recurente, acest model extrage caracteristici spatio-temporale din subsecvente ale aceluiasi video, prin convolutii, pe care le foloseste ca intrari pentru reteaua recurenta, la fiecare moment de timp. Figura 2 prezinta arhitectura acestui model. In acest caz, antrenarea celor doua retele se face separat. Pentru a fi antrenabile individual, ambele au nevoie de un clasificator liniar care calculeaza probabilitatea intrarii de a apartine in clasele considerate.
Reteaua convolutionala in trei dimensiuni primeste la intrare o insiruire de 9 cadre consecutive de dimensiune 34x54, la fiecare moment de timp. Pentru cadrul curent, se aleg cele 4 cadre anterioare si urmatoarele 4 cadre pentru a se contrui volumul ce va fi trimis la intrarea retelei. Straturile folosite in aceasta retea sunt cele special concepute pentru prelucrarea in trei dimensiuni si anume cele de convolutie - *Conv3D*, rectificare – implementat in Keras printr-un strat de tip *Lambda* si esantionare - *MaxPooling3D*. Reteaua contine doua blocuri formate din insiruirea celor trei straturi descrise si un al treilea nivel de convolutie urmat de clasificatorul liniar clasic alcatuit din 2 straturi de tip *Dense* complet conectate.
Primul strat de convolutie, foloseste 7 filtre de dimensiune 7x7x5, generand astfel 7 volume de dimensiune 28x48x5. Straturile de rectificare si esantionare nu contin parametrii antrenabili, scopul fiind doar de a modifica intrarea prin aplicarea operatiei de modul in cazul primului si prin reducerea

dimensiunii spatiale la jumatate in cazul celui de-al doilea. Functia modul trebuie aplicata asupra intrarii folosind un obiect de tip *Layer* care permite propagarea erorii inapoi in retea in faza de *backpropagation* si calculul derivatelor. De aceea, am folosit un strat de tip *Lambda* care permite o implementare personalizata in functie de nevoie. Folosind stratul de tip *MaxPooling3D* am implementat esantionarea extragand valoarea maxima din fiecare vecinatate de ordin 2.

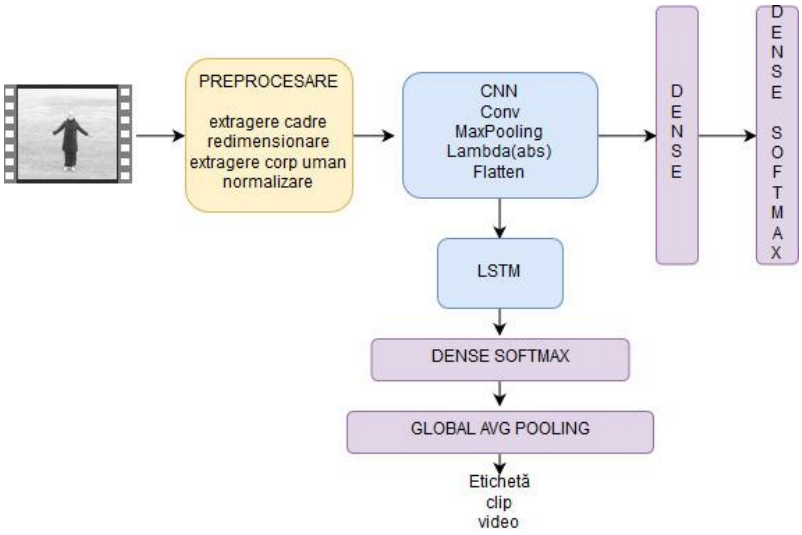


Figura 2. Arhitectura modelului format din convolutii 3D si recurente

Urmatoarele 3 straturi folosesc acelasi principiu si aceasi ordine ca cele descrise mai sus. Legaturile intre primul strat de esantionare si al doilea strat de convolutie se realizeaza intr-un mod diferit de cel clasic, al straturilor complet conectate. La iesirea stratului intai de convolutie exista 35 de volume obtinute prin convolutii cu filtre de dimensiune 3x3x3. Primele 14 dintre acestea rezulta din convolutia a doua filtre diferite asupra fiecarui volum din cele sapte gasite la iesirea stratului doi de esantionare. In plus, pentru fiecare pereche de volume, prin convolutia fiecarui volum cu un filtru diferit se obtin doua iesiri care se fuzioneaza, folosind functia de maxim. Toate aceste convolutii se realizeaza individual folosind stratul *Conv3D*, iar rezultatele se combina folosind stratul de concatenare *Maximum*. Ultimul strat de convolutie aplica cinci filtre de dimensiune 3x3x3 si este complet conectat la cele 35 de volume de dimensiune 10x20x3 obtinute in urma esantionarii prezentate anterior. Astfel la iesirea stratului, datele vor avea dimensiunea de 5x3x8x1, urmand a fi aplatizate si transformate intr-un vector unidimensional de 120 de caracteristici. Acest vector va fi folosit atat pentru antrenarea acestei retele mai departe prin clasificator, dar reprezinta si intrarea retelei recurente. Reteaua contine si un clasificator liniar implementat folosind doua straturi de tip *Dense*, complet conectate. Primul strat contine 50 de neuroni, iar stratul final, cel care produce iesirea prezinta un numar de neuroni egal cu numarul de clase prezise. In mod natural, se foloseste functia de activare *softmax* pentru a calcula probabilitatea de apartinere a subsecventei in fiecare clasa.

Reteaua recurenta de tip Long Short Term Memory primeste spre procesare un vector de atribute invatat de reteaua convolutionala, la fiecare moment de timp. Din aceasta cauza, datelor ce trebuiesc servite retelei trebuie sa se regaseasca sub forma unui vector de doua dimensiuni: numarul de cadre extrase din video si numarul de atribute. Reteaua convolutionala poate sa produca date de dimensiunea mentionata daca se foloseste un strat auxiliar aplicat asupra fiecarui strat din reteaua originala denumit TimeDistributed. Dimensiunile clasice ale datelor folosite intr-o retea convolutioanala in trei dimensiuni sunt pastrate, diferenta este ca acest supra-nivel forteaza reteaua sa prelucreze toate cadrele primite ca un intreg si astfel iesirea fiecarui strat va contine o dimensiune in plus si anume numarul de cadre.

Monitorizarea calitatii somnului – Parametrii de somn au fost colectati folosind senzorul Emfit QS[1]. Datele preluare de la senzorul Emfit QS (prin intermediul Cloud-ului Emfit) sunt analizate in vederea identificarii diferitelor corelari intre calitatea somnului si parametri masurati in perioada de somn. Acesta masoara urmatorii parametri pe durata somnului:

- ritmul cardiac
- rata de respiratie
- miscarile din timpul somnului
- numarul de treziri si ridicari din pat
- durata somnului (nu include timpul in care utilizatorul a fost treaz)
- activitatea sistemului nervos

si calculeaza urmatoarele elemente referitoare la perioada de somn (aceste valori sunt calculate in interval de 10-15 minute dupa finalizarea perioadei de somn; o perioada de somn este finalizata daca senzorul nu mai detecteaza prezenta utilizatorului in pat pentru o perioada mai mare de 10 minute; orice parasire a patului pentru o perioada mai mica de 10 minute este contorizata sub forma unei treziri in timpul somnului):

- scorul asociat perioadei de somn, calculat conform:
- variatia ritmului cardiac
- perioadele de somn (impreuna cu durata asociata).

Clasele de somn detectate de sensor sunt REM, Light, Deep. De asemenea se consider si perioadele in care utilizatorul este treaz (AWAKE).

In cadrul acestei etape s-a continuat analiza datelor pe baza modelului de regresie liniara utilizat in etapa anterioara, totodata evaluandu-se alte modele: regresie polinomiala, Support Vector Regressor (SVR) si Random Forest.

Modele folosite:

- Regresia liniara: este folosita pentru a gasi o relatie intre doua sau mai multe variabile. De obicei, exista o variabila care este dependenta de toate celelalte, acestea purtand numele de variabile independente. Relatia dintre variabila dependenta si restul variabilelor nu este una determinista, ci una statistica. Ideea de baza in cadrul folosirii modelelor bazate pe regresie liniara este de a gasi o functie care modeleaza cel mai bine setul de date. Pentru a se observa cat de bine modelul invata un set de date, se defineste eroarea ca fiind distanta dintre valoarea prezisa de model in acele puncte si valoarea reala. In cazul modelelor care folosesc gresie liniara, functia este o combinatie liniara a variabilelor independente si consta intr-o lista de coeficienti, fiecare reprezentand ponderea variabilei in cadrul functiei.
- Regresia Polinomiala: este o modalitate de a modela relatia dintre variabila dependenta si cea independenta prin intermediul unui polinom de grad N. Aceasta relatie nu este una liniara in functie de variabila dependenta, folosindu-se pentru a modele dependete neliniare dintre perechi de variabile. Aceasta metoda de regresie ramane totusi una liniara cand vine vorba de coeficientii polinomului care trebuie gasiti, fiind, de altfel, parametrii modelului.
- Support Vector Regressor (SVR): pleaca de la un set de date de antrenare ce contine N puncte si o valoare eps si incearca sa gaseasca o functie, astfel incat, distanta dintre valoarea prezisa de functie si valoarea reala sa fie cel mult \textit{epsilon} pentru orice exemplu din setul de antrenare. Acest lucru este foarte important atunci cand, pentru valoarea prezisa, se admite o eroare de cel mult eps[2].
- Random Forest: este un model aditiv, care combina o secventa de modele de baza numite arbori de regresie. Modelul are doua etape: partitia spatiului si gasirea unui model simplu pentru fiecare partitie. Arborii de regresie folosesc arbori de decizie pentru partitia spatiului si, de asemenea,

[1] https://www.emfit.com/

[2] https://alex.smola.org/papers/2003/SmoSch03b.pdf

sunt arbori de decizie cu valori continue. Frunzele corespund unor regiuni invecinate din spatiul de intrare. Arborii de regresie sunt o metoda de regresie neliniara, deoarece folosesc partitia spatiului

Analiza datelor: in vederea stabilirii perechilor care se afla in stransa legatura. Agregarea informatiei a presupus urmatoarele lucruri:

- adunarea datelor referitoare la rezumatele perioadelor de somn
- esantionarea valorilor medii pentru semnele vitale (ex: ritmul respirator, cardiac), tinand cont de intervalele in care au loc clasele de somn. Acest lucru a fost necesar pentru a se testa daca exista o repetare asemanatoare a claselor de somn si a semnelor vitale in anumite intervale
- crearea de fisiere ce contin valorile medii ale semnelor vitale (ex: ritmul respirator, cardiac) si durata diferitelor clase de somn. Acestea au fost necesare pentru a se testa daca durata totala a unei anumite clase de somn sau chiar durata totala a somnului este influentata de valorile medii ale anumitor semne vitale
- adunarea informatiei despre miscarea utilizatorului in timpul somnului si clasele de somn sau semnele vitale. Miscarile unei persoane pot oferi informatii despre cat de linistita este aceasta in timpul somnului. Dispozitivul Emfit QS ofera detalii despre activitatea persoanei din timpul somnului: inregistreaza miscari mai mari cauzate de bataile inimii sau miscarile membrelor si nivelul de agitatie

Analiza datelor s-a facut atat prin inspectarea coeficientului de corelatie Pearson:

$$\delta(a,b) = \frac{E(a,b)}{\sigma_{a^*}.\sigma_b} \quad (1)$$

unde

- a, b variabile aleatorii
- E(a, b) covarianta dintre cele 2 variabile
- $\delta(a,b)$ coeficientul Pearson
- σ_a , σ_b deviatiile standard

intre oricare doi parametri dintr-un data frame, cat si prin inspectarea scorului oferit de modelele incercate pe respectivele seturi de date.

In prima instanta, s-a incercat eliminarea perechilor de parametri, care aveau un coeficient de corelatie foarte mic, deoarece acestea nu se influentau reciproc (Figura 3)

```
def get_correlation(data_frame):
    """Creeaza un dictionar in care
    cheia este un tuplu (coloana1, coloana2),
    iar valoarea este coeficientul de corelare
    intre valorile acelor coloane

    Keyword arguments:
    data_frame -- data frame-ul aferent fisierului CSV
    """

    col_correlations = data_frame.corr()
    col_correlations.loc[:, :] = np.tril(col_correlations, k = -1)
    cor_pairs = col_correlations.stack()

    return cor_pairs.to_dict()
```

Figura 3. Obtinerea ceoficientului de corelatie intre perechi de doua variabile

Dupa eliminarea perechilor care nu prezentau o corelatie semnificativa, s-a inceput antrenarea diferitelor modele, fiind modificati atat parametrii reprezentativi pentru modelul respectiv, cat si numarul de date

din setul de antrenare. S-au pastrat modelele care au oferit cele mai mari scoruri pe seturile de date de testare. Pentru a se vedea ce parametri ofera cel mai mare scor pe datele de test, s-a utilizat *GridSearchCV*[3] din *Python*.

Au fost incercate mai multe modalitati de antrenare precum: gasirea unui numar de zile care rezulta in modelul cu cel mai bun scor pe zile ramase (aceasta metoda a dat cele mai bune rezultate), antrenarea pe un anumit numar de zile si adaugarea in setul de antrenare a acelor puncte care prezentau o diferenta semnifactiva fata de valoarea prezisa (acest lucru s-a dovedit a fi o abordare care strica modelele, deoarece acestea tind sa invete valori marginale si sa se departeze astfel de valorile obisnuite ale utilizatorilor) si antrenarea pe diferite permutari generate pe baza setului total de date cu scopul de a gasi un subset de zile din care modelul sa poata sa invete evolutia acestor parametri (acesta metoda a fost destul de costisitoare, dar nu a produs rezultate mai bune decat prima metoda, deoarece existau seturi de date care contineau puncte foarte apropiate si in aceste cazuri modelul invata evolutia acelui subset si obtinea scoruri mici pe subsetul ramas din permutarea generata).

Metodele enumerate anterior au fost incercate pe perioade in care utilizatorul dormea cel putin doua ore, de altfel, aceasta fiind si durata minima care a fost considerata ca fiind o perioada de somn pentru utilizator. Perioadele de somn care aveau durata sub aceasta valoare nu au fost luate in considerare.

Pentru a se varia doar numarul de zile care este folosit pentru invatare, s-a utilizat *PredefinedSplit*[4] pentru a crea o impartire prestabilita de indecsi la fiecare pas dupa o regula conform documentatiei. Mostrele date care sunt folosite la testare vor avea pe pozitia lor din setul de date valoarea -1, iar cele folosite pentru validare vor avea pe pozitia lor din setul de date valoarea 0. Astfel, se va varia numai numarul de zile ce intra in componenta setului de date de invatare, pentru a putea simula procedeul de invatare al comportamentului utilizatorului.

Pentru fiecare model utilizat, se intoarce cel mai bun scor, cea mai buna combinatie de parametri, numarul primelor zile folosite pentru antrenarea acestuia si o lista ce contine evolutia celui mai bun scor in functie numarul primelor zile sunt folosite in setul de date de antrenare.

La finalul analizei unei perechi, s-a salvat un fisier *JSON*, ce contine cele mai bune modele din cele patru incercate (Figura 4).

Fisierul contine pentru fiecare model cel mai bun scor obtinut, numarul primelor zile folosite la antrenare si alti parametri reprezentativi, astfel incat sa se poata reconstrui modelul:

- pentru modelul *Regresie Polinomiala*, s-a pastrat gradul care a condus la cel mai bun scor
- pentru modelul *Random Forest*, s-a pastrat un dictionar ce contine adancimea maxima, numarul de estimatori folositi, starea aleatoare de la care s-a plecat, numarul minim de mostre pentru ca un nod sa poata fi considerat frunza, precum si un parametru care indica daca trebuie folosit tot setul de date sau diferite mostre de date din set pentru constructia arborilor
- pentru modelul *Support Vector Regressor*, s-a pastrat tipul de kernel si parametrii care descriu kernel-ul respectiv

Metoda de validare pentru modelele obtinute este de a compara scorurile pe ultimele zile din setul de date, scopul fiind acela ca modelele sa invete profilul utilizatorului pe o perioada continua de timp, iar dupa aceea sa incerce sa prezica valorile pentru variatiile din zilele ce vor urma.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.PredefinedSplit.html

```
{
    "forest": {
        "score": 0.9783579340334506,
        "best_params": {
            "bootstrap": true,
            "max_depth": 5,
            "min_samples_leaf": 2,
            "n_estimators": 4,
            "random_state": 8
        },
        "train_size": 53
    },
    "poly": {
        "score": 0.9813587860533358,
        "best_params": {},
        "train_size": 47,
        "degree": 3
    },
    "linear": {
        "score": 0.9799508783098683,
        "best_params": {},
        "train_size": 39
    },
    "svr": {
        "score": 0.9815234543042574,
        "best_params": {
            "C": 4,
            "coef0": 4,
            "degree": 3,
            "epsilon": 0.01,
            "gamma": 1,
            "kernel": "poly"
        },
        "train_size": 47
    }
}
```


Figura 4. Exemplu de fisier JSON ce contine cele mai bune modele

Activitatea 3:2: Contributii la realizarea sistemului integrat: in cadrul acestei activitati a fost dezvoltata interfata ce permite monitorizarea parametrilor de somn, cu evidentierea zilelor in care acesti difera fata de modelul invatat, precum si integrarea unui set comenzi vocale (in limba engleza) pentru interfata sistemului IONIS.

Interfata componentei de monitorizare a parametrilor somnului: Pentru fiecare pereche de parametri, este creat un model per utilizator. Aplicatia se deschide cu o fereastra care cere selectarea utilizatorului, pentru care se doreste sa se inspecteze parametrii de somn.

Utilizatorii pentru care exista un istoric de monitorizare vor fi afisati in acest ecran, iar cel care foloseste aplicatia, va putea selecta persoana pe care vrea sa o analizeze prin intermediul unui combo box (Figura 5). S-a ales folosirea unui combo box pentru a se putea adauga oricati utilizatori fara a exista necesitatea unui spatiu suplimentar in interfata pentru utilizatorii introdusi ulterior. Singura necesitate, din punct de vedere al spatiului, pe care o necesita aceasta fereastra de deschidere este cea legata de desfasurarea listei din combo box pentru a se putea selecta utilizatorul dorit. Figura 6 prezinta perechile de parametri care pot fi analizate valori reale – raportate la modelul invatat.

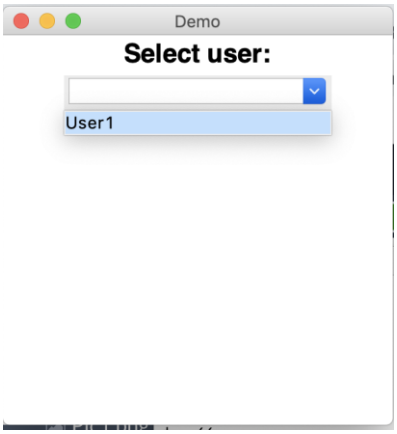


Figura 5. Selectie utilizator

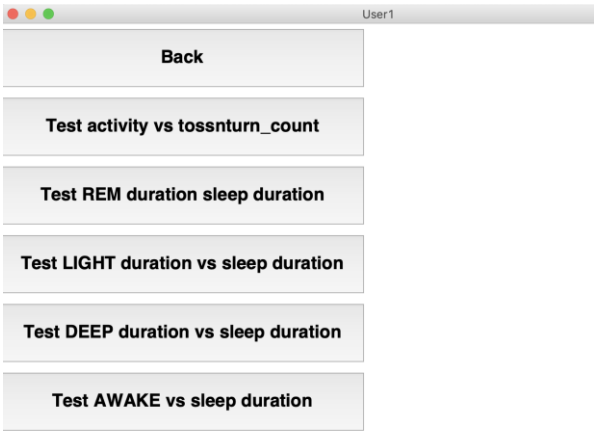


Figura 6. Vizualizare profil - perechi de parametri

Dupa cum se poate vedea in Figura 7, utilizatorul a ales sa vada istoricul de somn pentru ultimele zece zile, pentru perechea de parametri data de durata de somn si durata in stadiul LIGHT, iar apoi a ales sa inspecteze pulsul mediu, rata medie de respirati, durata de somn si numarul de treziri avute in aceste 10 zile (Figura 8). Aplicatia a generat primul grafic in care sunt coordonatele punctelor si dreapta de regresie descrisa de modelul invatat pe setul de date de la utilizatorul respectiv. Al doilea grafic este graficul cu bare, in care apar valorile reale si cele prezise de model. Barele din graficul 2 din figura Figura 7 care au conturul marcat cu negru evidentiaza zilele in care diferentele dintre valoarea reala si cea prezisa sunt mai mari cu 20% decat valoarea prezisa.

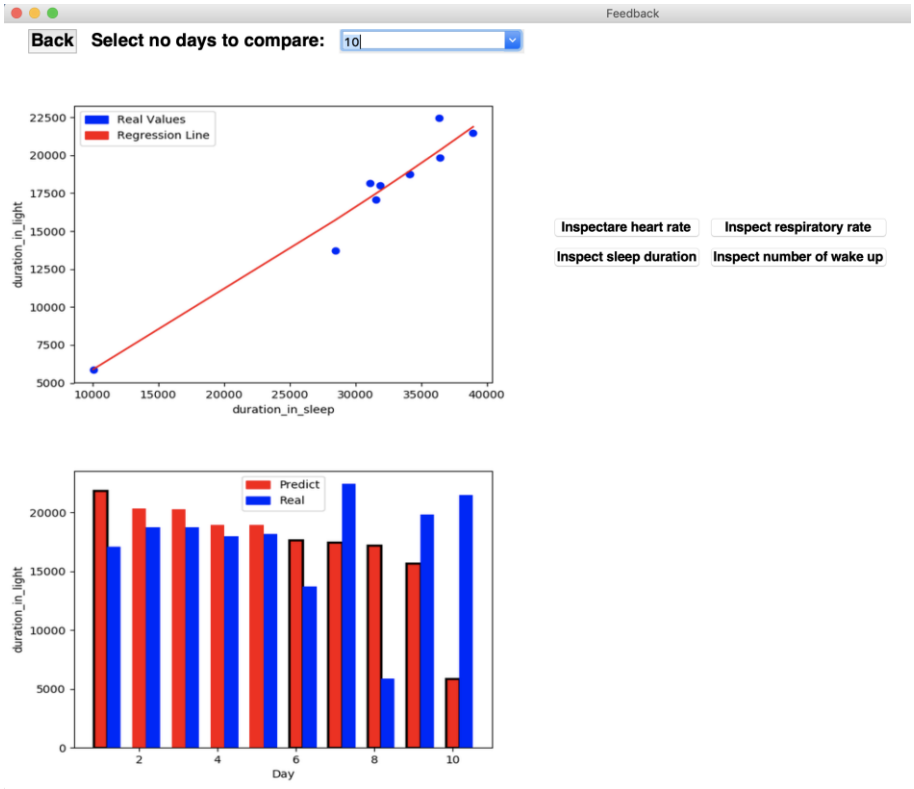
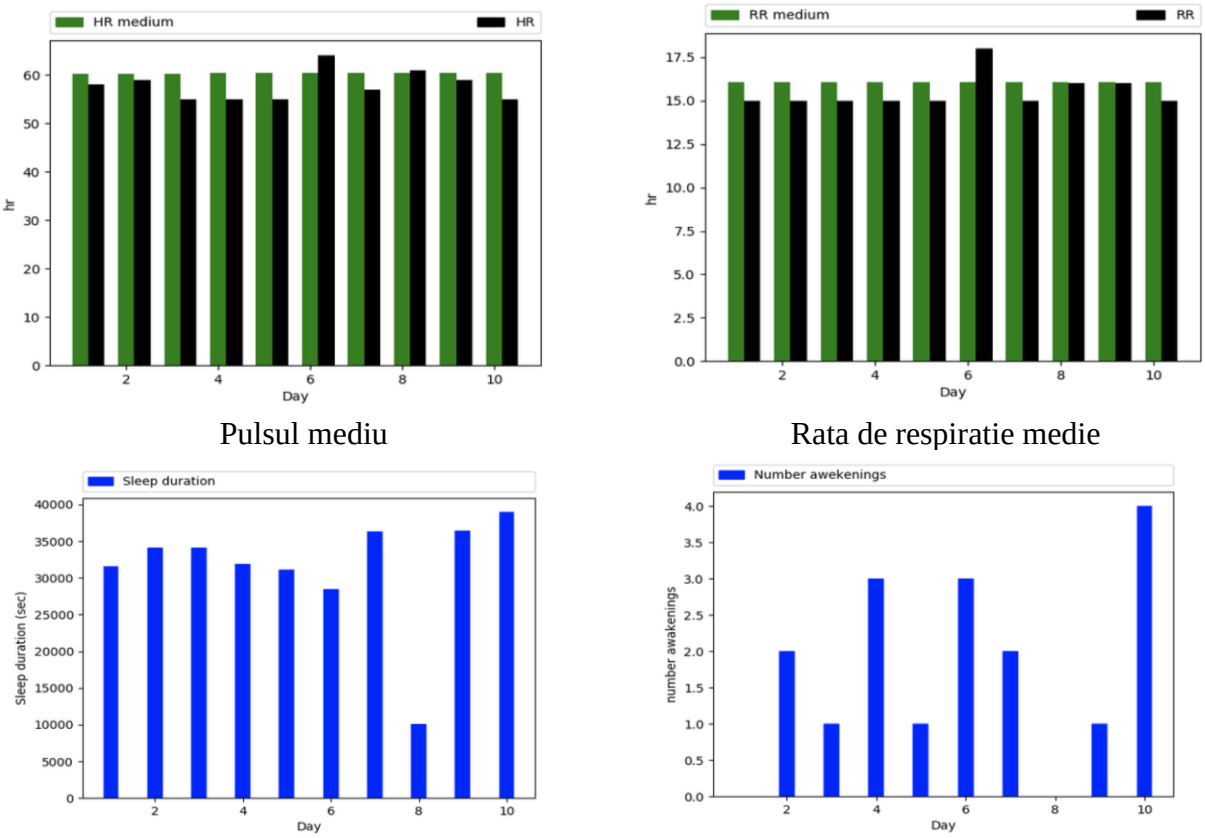


Figura 7. Vizualizare valori reale vs model invatat

Dupa cum se poate observa din capturile de ecran aplicatiei, utilizatorul interactioneaza la orice moment numai cu o singura fereastra si are la dispozitie un buton "Back" prin care acesta poate sa revina la fereastra anterioara. Acest lucru a fost necesar, deoarece in biblioteca *Tkinter*[5], in care este realizata interfata, daca exsita mai multe ferestre in acelasi timp, exista posibilitatea ca pe anumite versiuni de *Python* sa se piarda imaginile sau graficele. Astfel, s-a ales acest mod de interactiune cu utilizatorul pentru a avea cate o fereastra cu care acesta sa interactioneze in fiecare moment.



Pulsul mediu

Rata de respiratie medie

Durata de somn

Numar de treziri

Figura 8. Inspectare parametri pentru ultimele 10 zile

Interfata sistemului IONIS: Platforma IONIS vizeaza in principal persoanele in varsta, astfel ca proiectarea interfetei a luat in considerare nevoile si cerintele speciale ale acestei categorii de utilizatori, cum ar fi fontul si dimensiunea butoanelor (mari), navigare simplificata, sprijinirea unor modalitati mai naturale de interactiune intre sistem si utilizatori. Interfata integreaza diferite pictograme pentru a usura navigarea si interpretarea informatiilor pentru utilizator, asa cum este ilustrat in Figura 9.

Pictogramele utilizate sunt usor de interpretat. Interfata va accepta comenzi vocale - in limba engleza. Utilizatorul poate accesa interfata de pe orice dispozitiv, datele sale de sanatate care reflecta valorile in timp real, pentru orice perioada specificata, precum si diferite rapoarte referitoare la acesti parametri. Utilizatorul poate vizualiza datele medicale anterioare in diferite vizualizari de exemplu sub forma de grafic, asa cum este ilustrat in Figura 10.

Pentru intelegeea comenzilor vocale, a fost utilizata biblioteca RASA pentru limba engleza. Aceasta primeste la intrare comanda utilizator intr-un format text, rezultat din componenta de recunoastere

[5] https://docs.python.org/2/library/tkinter.html

automata a vorbirii. La iesire, se genereaza un fisier JSON care contine informatii utile cu privire la comanda utilizatorului (intentie, entitati, etc) asa cum este ilustrat in Tabel 1.

Figura 9. Pagina principala a interfetei.

Figura 10. Afisarea masuratorilor pentru greutate (sub forma de grafic).

Intrare RASA: How will be the weather tomorrow.

Rasa Output (partial):
{
"intent": "get_weather",
"entities": {
"location": "the_current_location_of_the_user",
"datetime": "tomorrow"
}
}

Tabel 1. Exemplu intrare – iesire RASA.

Au fost create intentii diferite, fiind associate diferite entitati, fiecarei intentii. In exemplul ilustrat in Tabel 1, textul de iesire creat de ASR (Automatic Speech Recognition) „ How will be the weather tomorrow?”, RASA recunoaste „get_weather” ca intentia rostirii si recunoaste doua entitati: o entitate „locatie” care reprezinta „the_current_location_of_the_user”, preluata din sistemul GPS al utilizatorului si entitatea „datetime” care este „tomorrow”. Fisierul JSON generat contine, de asemenea, informatii suplimentare, cum ar fi scorul asociat fiecarei recunoasteri. In plus, au fost create si utilizate pentru a antrena modelele de gestionare a dialogului RASA diferite succesiuni de interactiuni. Tabel 2 ilustreaza o astfel de secventa de interactiuni. De asemenea, a fost creat si un set de actiuni care ar trebui sa fie

executate de sistem, actiunile sunt reprezentate de mesaje care ar vor fi generate de sistem si coduri personalizate care ar trebui sa fie executate de sistem.

## Story 1 *(story name)* 	
* greet *(recognized intent)*
  - utter_greet *(system answer/action)*
* get_weather *(recognized intent)*
  - utter_weather *(system answer/action)*
* goodbye *(recognized intent)*
  - utter_goodbye *(system answer/action)*

Tabel 2. Secventa de interactiuni RASA.

In exemplul ilustrat in Tabel 2, utilizatorul initiaza o conversatie cu boot-ul folosind o expresie de salut „*Hello*”. RASA recunoaste intentia utilizarii „*Hello*” si genereaza raspunsul „*utter_greet*”: „*Hello, how can I help you?”*. Utilizatorul intreaba boot-ul despre vreme folosind urmatoarea propozitie: „*How will be the weather tomorrow*”. RASA recunoaste intentia rostirii utilizatorului „*get_weather*” si genereaza raspunsul „*utter_weather*”: „*Tomorrow, the weather will be warm and sunny*”. Utilizatorul finalizeaza conversatia spunand: „*Thanks and good bye*”. RASA recunoaste intentia de a spune de la utilizator „*goodbye*” si genereaza raspunsul „*utter_goodbye*”: „*Good bye*”.

Pentru evaluare, 10 utilizatori au testat comenzile vocale din interfata folosind 25 de comenzi diferite. In total, 250 de comenzi vocale au fost evaluate, rezultatele obtinute sunt urmatoarele: procentul recunoasterii intentiei a fost de 79%, iar procentul recunoasterii entitatii a fost de 71%. Exemplificarea pentru recunoastere intent, entity(ies) si iesirea generata este prezentata in Tabel 3 pentru o serie de comenzi in limba engleza.

utterance	intent	entity(ies)	iesire
How will be the weather *tomorrow*	get_weather	location: *current location of the user* datetime: *tomorrow*	*Tomorrow*, the weather will be warm and sunny
How will be the weather in *London* on *Monday*	get_weather	location: *London* datetime: *next Monday*	*Next Monday*, the weather will be cold and rainy in *London*
What is my *health status*	get_health_status	health: *health status*	You are doing fine.
Display my *blood pressure*	display_health_param	health_param: *blood pressure*	Orders the system to display the blood pressure chart for the measurements obtained during the last 7 days and to display the

			last blood pressure measurement result.
How much have I *walked today*	get_health_param	health_param: steps datetime: *today*	*Today*, you have walked 4716 *steps*.

Tabel 3. Asociere utterance - intent, entity(ies) - iesire generata - pentru comenzi vocale in engleza

Activitatea 3.3: Analiza datelor colectate in experimente realizate pe teren: in cadrul acestei etape au fost analizate metodele implementate in cadrul activitatii 3.1 pe date colectate de la utilizatorii sistemului IONIS.

Evaluarea recunoasterii activitatilor zilnice

Modelele de recunoastere a activitatilor umane au fost evaluate pe setul de date PRECIS HAR – set de date format din videouri ce contin atat cadre RGB cat si de adancime. Setul de date este compus din 16 activitati (stand up, sit down, sit still, read, write, cheer up, walk, throw paper, drink from a bottle, drink from a mug, move hands in front of the body, move hands close to the body, raise one hand up, raise one leg up, fall from bed and faint), executate de 50 de persoane - in total fiind 800 de secvente video cu cadre atat RGB cat si de adancime. Inregistrarea activitatilor s-a realizat in cadrul UPB. Retelele au fost antrenate folosind numai cadrele RGB, utilizand biblioteca Keras din Python in mediul de rulare Google Colaboratory. In urma evaluarii celor doua modele, acuratetea retelei convolutionale 2D a fost de 82%, iar cea pentru reteaua 3D a fost de 85%.

Evaluarea modelelor invatate folosind datele colectate de la senzorul Emfit QS.

Pentru gasirea si testarea perechilor de marimi, au fost folositi trei utilizatori: utilizator1, utilizator2 si utilizator3. Un aspect demn de mentionat este referitor la faptul ca cei trei utilizatori folositi au obiceiuri de somn diferite:

- utilizator1 doarme cel mai mult, destul de des, chiar si in timpul zilei cel putin o ora. In jurul noptii, durata minima de somn este de 6 ore si 33 de minute, iar cea maxima de 12 ore si 12 minute, majoritatea duratelor de somn fiind de cel putin 8 ore.
- utilizator2 doarme cel mai putin, avand un obicei dezordonat din punct de vedere al perioadelor de somn. Acesta are perioade dese in care doarme in jur 1-2 ore, se trezeste si sta treaz aproximativ o jumatate de ora dupa care adoarme inapoi. Durata minima de somn este de 2 ore si 35 de minute, iar cea maxima este de 9 ore si 38 de minute, majoritatea duratelor de somn fiind in jurul valorii de 6 ore.
- utilizator3 este cel care are cea mai constanta perioada de somn. Durata minima de somn este de 5 ore si 38 de minute, iar cea maxima de 12 ore, majoritatea duratelor de somn fiind in jurul valorile de 6-8 ore.

Datele folosite de la cei trei utilizatori au fost colectate pe parcursul a sase luni, pentru a putea realiza o comparatie despre cum obiceiurile de somn afecteaza perechile de parametri inspectati.

Modelele gasite se bazeaza pe regresie liniara si au fost obtinute din arbori de regresie, regresie liniara simpla, regresie polinomiala si vectori suport pentru regresie. Astfel, dupa analiza seturilor de date, au fost gasite corelatii intre diferite perechi de parametri, care descriu un utilizator din perspectiva duratelor de somn si din perspectiva activitatii sale din timpul somnului. Pentru fiecare utilizator s-a pastrat cel mai bun model, in functie de scorul pe setul de date de testare.

Perechile de de parametri care descriu un utilizator din perspecitva duratelor de somn sunt urmatoarele:

- durata petrecuta in stadiul REM si durata de somn
- durata petrecuta in stadiul de LIGHT si durata de somn
- durata petrecuta in stadiul de DEEP si durata de somn
- durata petrecuta in stadiul AWAKE si durata de somn (durata de somn nu influenteaza direct aceasta etapa)
- activitatea si numarul de miscari din timpul somnului.

Exemple de modele antrenate pentru perechile de parametri sunt descrise in Figura 11, Figura 12, Figura 13, Figura 14 si Figura 15:

a) Activitatea vs numarul de miscari din timpul somnului.
b) Durata light vs durata totala de somn.
c) Durata rem vs durata totala de somn.
d) Durata deep vs durata totala de somn.
e) Durata awake vs durata totala de somn.

Activitatea vs numarul de miscari din timpul somnului: Miscarile fine ale unui utilizator (activitatea) sunt definite ca fiind acele miscari mici, precum cele ale muschilor sau miscari superficiale ale membrelor, in timp ce miscarile mai proeminente sunt caracterizate ca fiind intoarceri. Atunci cand una dintre marimi creste, aceasta crestere are influenta si asupra celeilalte. O crestere a acestor valori pe o perioada indelungata poate indica un somn nelinistit.

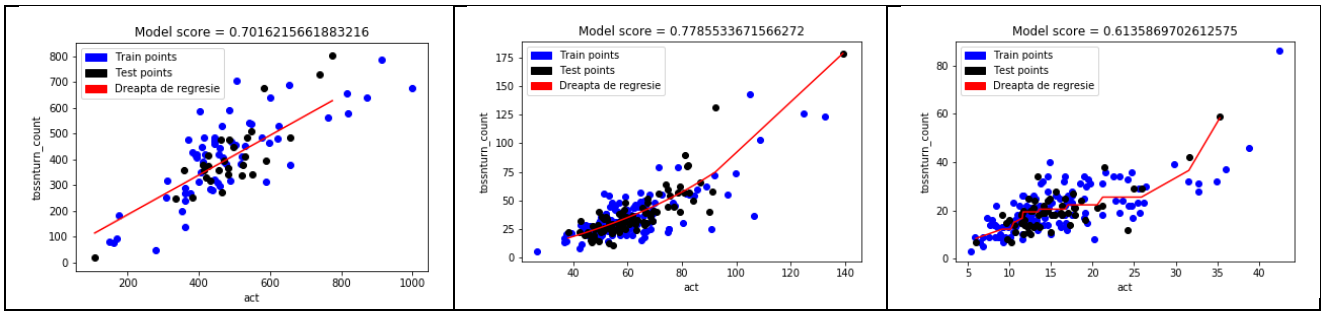


Figura 11. Activitatea vs numarul de miscari din timpul somnului

Durata light vs durata totala de somn: Stadiul LIGHT este descris ca fiind acea etapa de tranzitie intre starea AWAKE si starea DEEP. Pe pagina dispozitivului Emfit QS, este indicat ca aceasta durata sa fie intre 50% si 60%. S-a insepctat influenta duratei de somn asupra acestei etape pentru toti cei trei utilizatori si s-a constatat ca durata de somn LIGHT respecta in proportii foarte mari pragurile mentionate mai sus. Dintre toate cele patru modele incercate, rezultatul cel mai bun a fost obtinut prin folosirea vectorilor suport pentru regresie pentru toti cei trei utilizatori.

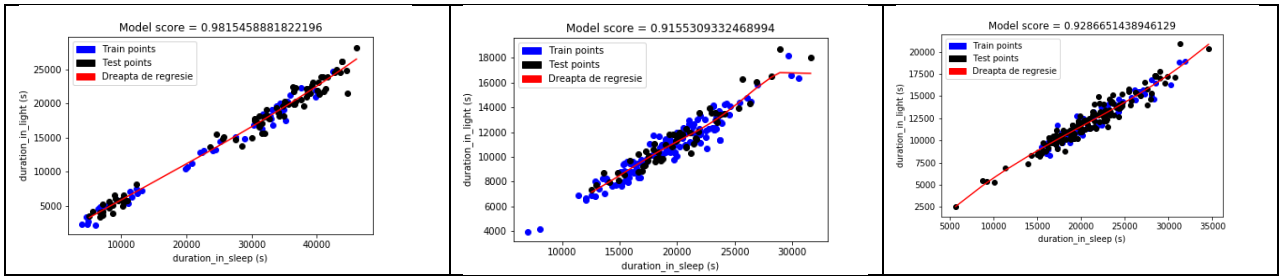


Figura 12. Durata light vs durata totala de somn

Durata rem vs durata totala de somn: Stadiul REM este acea faza a somnului caracterizata de o miscare rapida a ochilor, paralizie a muschilor si o variatie atat a presiunii arteriale cat si a ritmului inimii. Pe pagina dispozitivulu Emfit QS este indicat ca aceasta durata sa fie intre 20% si 25% pentru o recuperare completa. S-a insepctat influenta duratei de somn asupra acestei etape a somnului pentru toti cei trei utilizatori si s-a constatat ca in cazul utilizatorului 1 si a utilizatorului 3, cei care dorm mai mult, acuratetea este mai mare, iar in cazul utilizatorului 2, al carui obicei de somn este foarte variat, aceasta durata nu respecta in totalitate intervalul specificat mai sus.

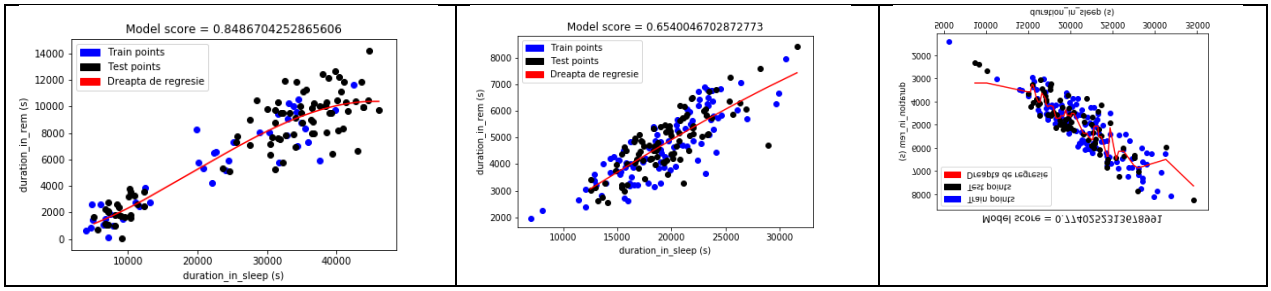


Figura 13. Durata REM vs durata totala de somn

Durata deep vs durata totala de somn: Stadiul DEEP este acea faza a somnului caracterizat de tonusul muscular aproape inexistent si un nivel scazut si constant al ritmului respirator si al presiunii arteriale. Aceasta etapa este esentiala pentru recuperarea fizica. Pe pagina dispozitivului Emfit QS este indicat ca aceasta durata sa fie intre 10% si 20% pentru o recuperare completa. S-a insepctat influenta duratei de somn asupra acestei etape a somnului pentru toti cei trei utilizatori si s-a constatat ca in cazul utilizatorului 1 acuratetea este mai mare, iar in cazul utilizatorilor 2 si 3, care dorm mai putin, aceasta variaza destul de mult.

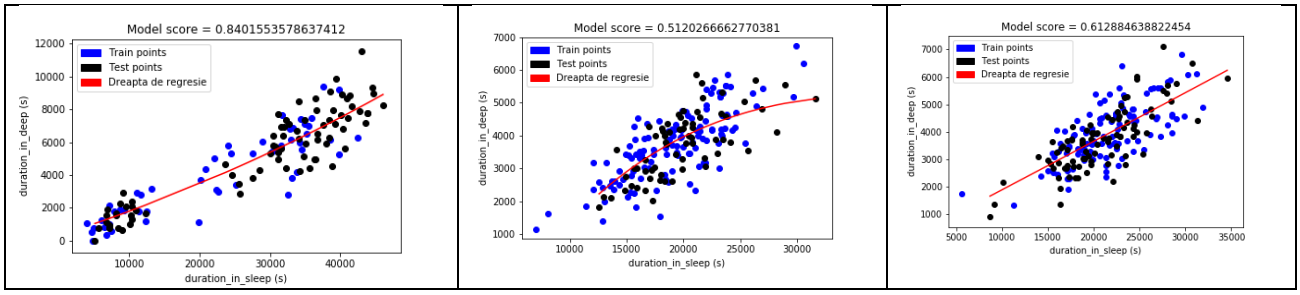


Figura 14. Durata deep vs durata totala de somn

Durata awake vs durata totala de somn: Stadiul AWAKE reprezinta etapa in care utilizatorul este treaz. Durata de somn nu are o influenta directa asupra acestei marimi, modelele obtinute avand o acurate scazuta. Valorile pentru aceasta marime nu sunt constante de la noapte la noapte, fiind foarte de greu de invatat fara a avea un istoric al activitatilor cotidiene realizate de utilizator.

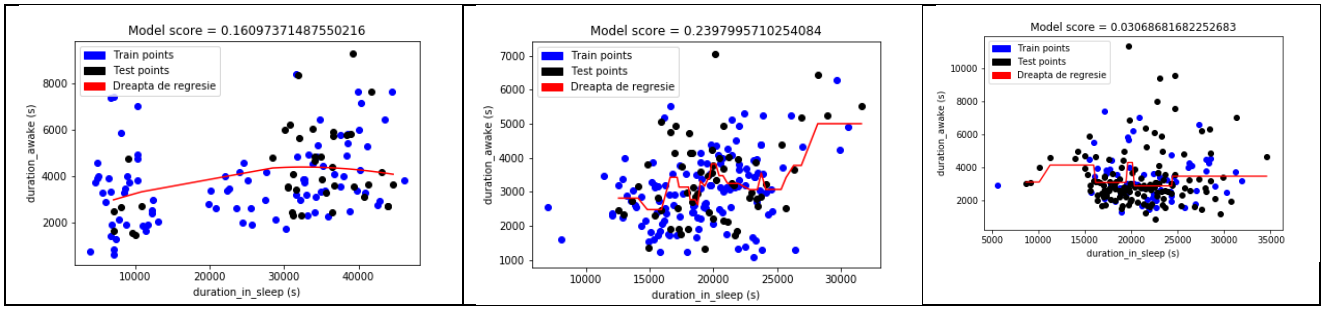
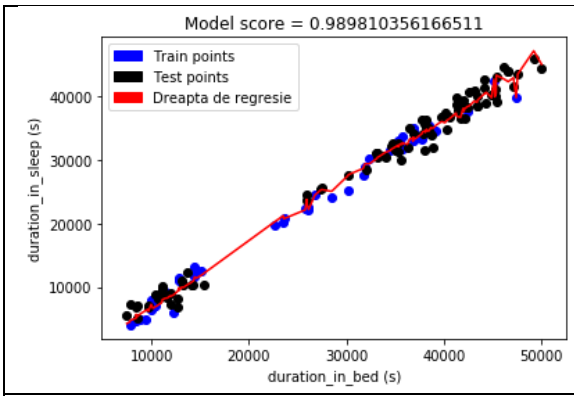
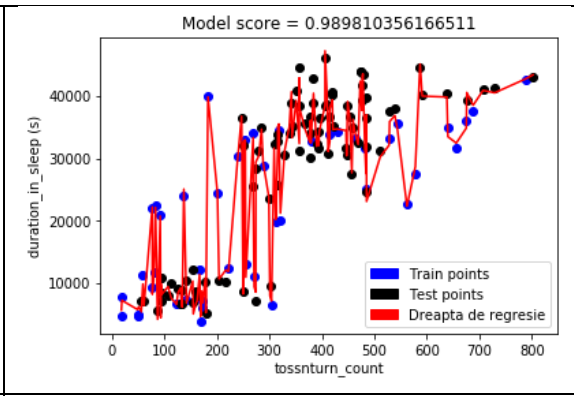


Figura 15. Durata awake vs durata totala de somn

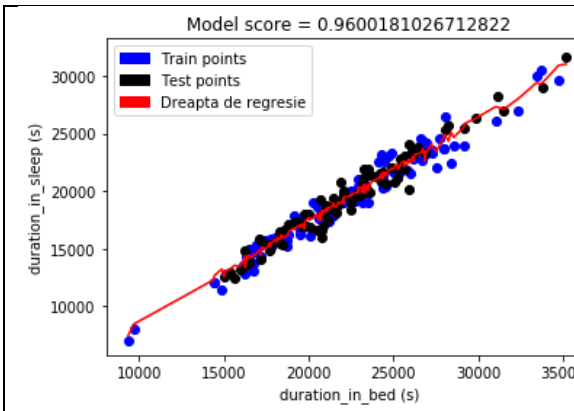
De asemenea a fost analizata durata in care utilizatorul a stat in pat, nivelul de agitatie si durata de somn. Durata in care utilizatoul sta in pat are o influenta ridicata asupra duratei totale de somn. Daca un utilizator, pe parcursul noptii, sta foarte mult in pat, iar durata sa de somn este foarte mica, atunci se poate spune ca acesta nu are un somn tocmai linistit sau are insomnii. S-a incercat un model de regresie cu trei variabile pentru a imbunatati acuratetea modelului (1). Prin adaugarea numarului de intoarceri ca variabila independenta, acuratetea a crescut cu 6% pentru utilizatorul 1 si cu 3% pentru ceilalti doi utilizatori. Figura 16 prezinta diferite modele pentru perechi de parametri: modelul invatat: antrenare si testare.



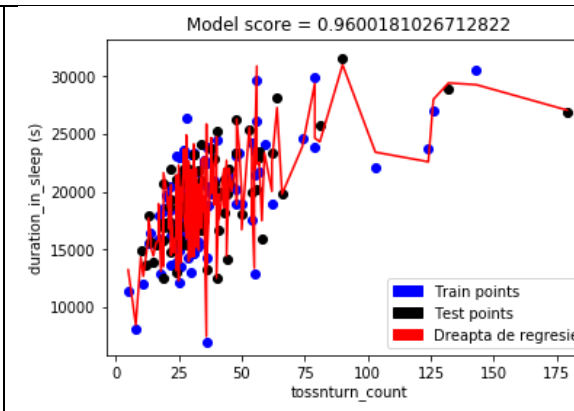
Durata de somn fata de durata in pat pentru utilizatorul 1 (Polinomial Grad 4)



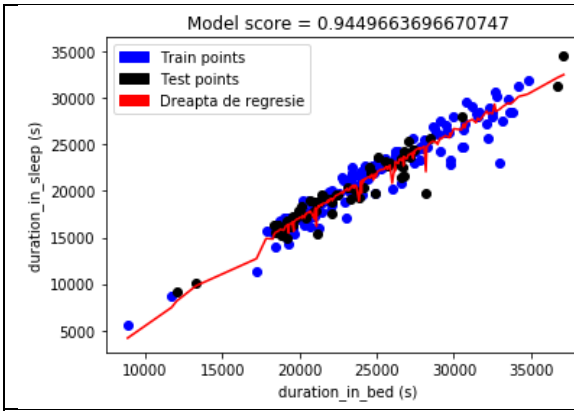
Durata de somn fata de nivelul de agitatie pentru utilizatorul 1 (Polinomial Grad 4)



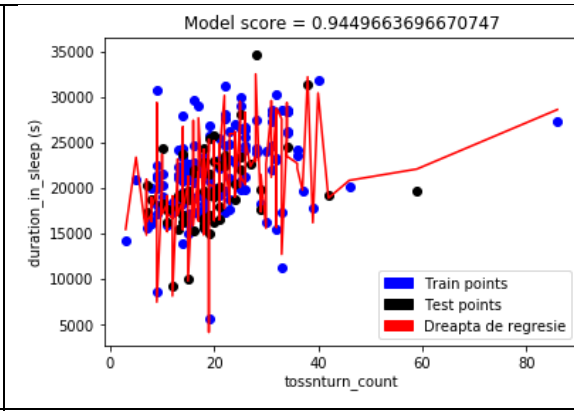
Durata de somn fata de durata in pat pentru utilizatorul 2 (SVR)



Durata de somn fata de nivelul de agitatie al utilizatorului 2 (SVR)



Durata de somn fata de durata in pat pentru utilizatorul 3 (SVR)



Durata de somn fata de nivelul de agitatie pentru utilizatorul 3 (SVR)

Figura 16. Exemple de modele

Discrepante intre valoarea prezisa de model si valoarea reala apar in cazul utilizatorului 2, care nu are un obicei regulat de somn, observandu-se ca durata in REM variaza destul de mult. In cazul celorlalti doi utilizatori, unde durata de somn ramane, in general, constanta, valoarea prezisa de model este aproape de valoarea reala.

Activitatea 3.4: Contributii la optimizarea platformei integrate: in cadrul activitatii s-a realizat fine tuning pentru modelele referitoare la reconasterea activitatilor zilnice, precum si analiza resurselor utilizate in cazul invatarii profilului utilizatorului bazat pe datele preluate din timpul somnului.

Recunoasterea activitatilor zilnice

Reteaua 2D a fost optimizata folosind principiul scaderii gradientului in mini-batch cu o rata de invatare de 0.01. Concepul de mini-batch se refera numarul de exemple folosite de retea pentru a face o modificare a ponderilor, in acest caz mai mare de 1 si mai mic decit numarul total de exemple. Aceasta valoare a fost determinata experimental, prin diferite incercari, am rulat cateva epoci si am ales valoarea in functie de cat de stabila se prezenta reteaua. M-a interesat sa nu apara fluctuatii bruste in functiile de eroare calculate pe setul de antrenare si pe cel de validare. Deoarece durata epocilor a fost mare, nu am putut sa fac teste pe diferite valori si configuratii ale parametrilor pe un numar mai mare de 10 epoci. Functia de eroare folosita a fost co-entropia, adaptata pentru mai multe categorii – categorical_crossentropy din Keras.

Monitorizarea somnului

Aplicatia utilizeaza foarte mult procesorul, deoarece biblioteca *scikit-learn* nu permite folosirea placii video din considerente de compatibilitate. Pe parcursul antrenarii modelelelor, utilizarea procesorului ajunge la 100%. Introducerea acestei functionalitati ar introduce diferite dependente in cod. Motivatia dezvoltatorilor acestei biblioteci este aceea ca *scikit-learn* se doreste a fi o biblioteca usor de instalat pe o varietate cat mai mare de platforme. Un alt scop pentru care nu se doreste introducerea suportului pentru placi grafice este acela ca acestea joaca un rol important doar in contextul retelelor neuronale, iar cei ce mentin biblioteca sustin ca performanta poate fi atinsa prin aplicarea algoritmului cel mai potrivit pe fiecare set de date, nu prin folosirea placilor video. In ceea ce priveste consumul de resurse, cele mai utilizate sunt CPU-ul si memoria RAM. Aplicatia consuma foarte mult timp pe procesor si pana la 2 GB de memorie RAM in procesul de cautare exhaustiva a celei mai bune combinatii de parametri folosind *GridSearchCV*.

Activitatea 3.5: Participare la manisfestari stiintifice, vizite de lucru: in cadrul acestei activitati s-au realizat urmatoarele:

- **Elaborare articole manifestari stiintifice**:

1. Neja Samar Brencic, Marius Dragoi, Irina Mocanu, Tomasz Winiarski - Intelligent solutions for elderly care, International Conference on Digital Health Technologies (ICDTH), Hammamet, Tunisia, 9-11 Decembrie 2019, in curs de publicare, *Springer Proceedings*
2. Veronica Radu, Irina Mocanu - Generating 3D Objects using Convolutional Neural Networks, Sensors, in evaluare, *jurnal cotat Q1*
3. Alexandru Placinta, Irina Mocanu, Oana Cramariuc, Bogdan Cramariuc – Learning healthy lifestyle through sleeping behaviour, in evaluare, INTED 2020, in evaluare, *ISI Proceedings*
4. O. Cramariuc, I. Mocanu, K. Broczek, D. Krivec, J. Kolakowski, N. Samar Brencic, Z. Nagymáté, I. Nagy, A. Consoli – What can we learn from an ICT project dedicated to people living with dementia, INTED 2020, in evaluare, *ISI Proceedings*

- 2 intalniri de lucru ale proiectului 17-18 iunie Varsovia, Polonia, 9-10 noiembrie, Nova Gorica, Slovenia
- Prezentare poster in cadrul forumului AAL 2019, 23-25 septembrie 2019, Aarhus, Danemarca
- Prezentarea sistemului IONIS in cadrul conferintei Montessori Senior method in practice, 15-16 iunie 2019, Varsovia, Polonia
- Prezentarea sistemului IONIS in cadrul conferintei ASK 2019 - LETNA KONFERENCA O DEMENCI, 8-9 Noiembrie 2019, Nova Gorica, Slovenia

- **Intalniri priect**:
  - 17-18 iunie 2019 – Varsovia, Polonia, intalnire la care s-a analizat stadiul curent al proiectului, precum si stabilirea etapelor urmatoare din proiect;
  - 9-10 noiembrie 2019 – Nova Gorica, Slovenia, intalnire la care s-a analizat stadiul curent al proiectului si s-au planificat si discutat principalele aspecte ce trebuie avute in vedere pentru finalizarea sistemului IONIS

- **Prezentare poster descriere proiect in cadrul forumului AAL, 23-25 Septembrie 2019, Aarhus, Danemarca:** Indoor and outdoor support for people with mild cognitive impairment and memory loss (Figura 17).

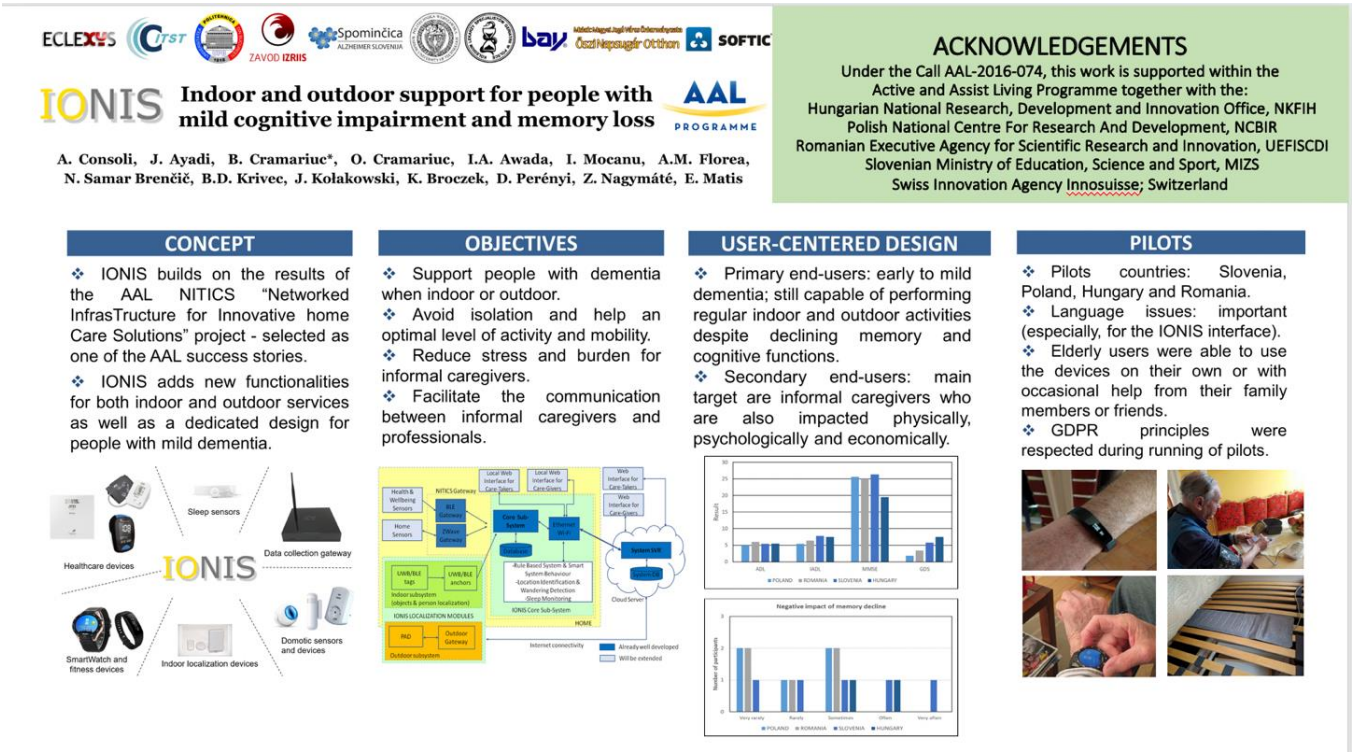


Figura 17. Poster AAL Forum 2019

5. Prezentare rezultate verificabile etapa

- Activitatea 3.1: Contributii la dezvoltarea modulelor sistemului
- Activitatea 3.2: Contributii la realizarea sistemului integrat
- Activitatea 3.3: Analiza datelor colectate in experimente realizate pe teren
- Activitatea 3.4: Contributii la optimizarea platformei integrate
- Activitatea 3.5: Participare la manifestari stiintifice, vizite de lucru

6. Concluzii

Obiectivele au fost realizate integral, gradul de atingere al rezultatelor fiind de 100%. In cadrul **activitatii 3.1** a fost implementata o metoda de recunoastere a activitatilor zilnice in vederea eliminarii detectiilor false ale ratacirilor (metoda de identificare a ratacirilor a fost prezentata in etapa 2), precum si extinderea metidei de extragere a informatiilor referitoare la corelea parametrilor corespunzatori tulburarilor de somn. **Activitatea 3.2** a constat in dezvoltarea interfetei sistemului IONIS folosind comenzi vocale. In cadrul **activitatii 3.3**, au fost testate metodele dezvoltate in cadrul activitatii 3.1 si 3.2. Analiza resurselor utilizate de metoda de identificare a profilului utilizatorului prin determinarea corelatiilor intre parametri de somn, precum si fine tuning pentru metoda de recunoastere a activitatilor umane au fost realizate in **Activitatea 3.4**. **Activitatea 3.5** a avut ca scop participarea la manifestari stiintifice, precum si vizite de lucru – in cadrul acestei etape a fost prezentat posterul ce descrie aspectele principale ale proiectului in cadrul forumului AAL (23-25 Septembrie 2019), Aarhus, Danemarca, au fost realizate 4 articole stiintifice: 1 articol acceptat in cadrul unei conferinte internationale cu proceedings indexat Springer, 2 articole in evaluare in cadrul unor conferinte cu proceedings indexate ISI si 1 articol in evaluare in jurnal cotat Q1. Proiectul IONIS a fost prezentat in cadrul a doua conferinte specifice problemelor dementei. Au avut loc 2 intalniri cu membrii consortiului (17-18 iunie 2019, Varsovia, Polonia, 9-10 noiembrie 2019, Nova Gorica, Slovenia).

7. Diseminare

- Elaborare 4 articole (1 lucrare in jurnal cotat Q1 si 3 lucrari la conferinte proceedings-uri indexate ISI si BDI).
- 2 intalniri de lucru ale proiectului 17-18 iunie Varsovia, Polonia, 9-10 noiembrie, Nova Gorica, Slovenia
- Prezentare poster in cadrul forumului AAL 2019, 23-25 septembrie 2019, Aarhus, Danemarca
- Prezentarea sistemului IONIS in cadrul conferintei Montessori Senior method in practice, 15-16 iunie 2019, Varsovia, Polonia
- Prezentarea sistemului IONIS in cadrul conferintei ASK 2019 - LETNA KONFERENCA O DEMENCI, 8-9 Noiembrie 2019, Nova Gorica, Slovenia

8. Pagina web a proiectului – actualizata cu datele ultimei raportari.

Pagina proiectului este: aimas.cs.pub.ro/ionis/