**Invatare Automata**

**Tema - Etapa 1**

**Lucian-Florin Grigore 343C4**

*Facultatea de Automatica si Calculatoare*

*Universitatea Politehnica, Bucuresti*

*Codul sursa al acestei lucrari poate fi gasit in Google Colab la* [*acest link*](https://colab.research.google.com/drive/15ubvponp44s44PHzkxiX5NsoJIRI8FAB?usp=sharing)*.*

**Cerinta 1. Exploratory Data Analysis**

**Frecventa de aparitie a claselor in setul de date pentru UWaveGesture**

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Observam ca setul de date UWaveGesture contine un numar egal de clase atat in setul de antrenare, cat si in cel de testare.

**Afisarea unei serii temporale pentru fiecare gest**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Pentru fiecare serie temporala am luat din setul de antrenare primul exemplu din clasa respectiva.

**Distributia valorilor per fiecare axa, per gest**

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Se poate observa ca majoritatea datelor se afla intr-un interval relativ restrans, ceea ce ar putea reprezenta o dificultate in antrenare, atunci cand se doreste optimizarea modelului si obtinerea unei acuratete foarte buna.

**Frecventa de aparitie a claselor in setul de date pentru UWaveGesture dataset**

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Observam ca setul de date PEMS-FS contine un numar inegal de clase atat in setul de antrenare, cat si in cel de testare. Cu toate acestea, nu se poate considera nicio clasa redundanta, toate avand un numar apropiat de exemple.

**Varierea ratei de ocupare pentru top 10 senzori cu deviatia cea mai mare pentru 8 zile selectate arbitrar uniform din totalul zilelor**

Chart

Description automatically generatedChart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

In urma analizei acestor grafice, cu mici exceptii, se observa cum majoritatea valorilor inregistrate de senzorii selectati se afla inspre limita inferioara a spectrului de valori. Asta poate reprezenta 2 lucruri:

1. Senzorii care au valori inregistrare mai mici (poate prin amplasarea lor in locatia respectiva) au si o sensibilitate la variatie, rezultand date mai imprastiate comparativ cu ceilalti senzori.
2. Posibil ca toti senzorii sa aiba valori mai apropiate de limita inferioara si atunci e nevoie de un pas in plus la preprocesarea datelor, dupa confirmarea acestei supozitii.

**Evolutia mediilor celor mai relevanti 10 senzori pe durata tuturor celor 440 de zile**

Chart

Description automatically generated

Putem remarca ca valorile in general sunt restranse intr-un interval mic de valori, apropiat destul de mult de origine.

**Cerinta 2.** *Pentru cerinta a doua am folosit datasetul UWaveGesture*

**Feature Selection**: Pentru a reduce datele de input la o dimensiune care poate fi gestionata si mai usor de analizat, am aplicat urmatoarele operatii:

* Am impartit fiecare axa (x, y si z) in ferestre de lungime 105 -> rezulta 3 ferestre per fiecare axa = 9 ferestre in total
* Pentru fiecare astfel de fereastra am facut media valorilor din seria de timp
* O intrare X din setul de date reprezinta aceste 9 valori obtinute in urma operatiilor de mai sus

In continuare, analiza atributelor si antrenarea modelelor este realizata pe aceasta noua reprezentare a datelor.

**Extragerea atributelor**

Applying mean on x\_axis: -1.1325396825710079e-07

Applying mean on y\_axis: -1.191991341994739e-07

Applying mean on z\_axis: -2.6096681093963078e-08

Applying std on x\_axis: 0.6537793630419304

Applying std on y\_axis: 0.7495870421412952

Applying std on z\_axis: 0.7088489128654653

Applying avg absolute diff on x\_axis: 0.5317972085502645

Applying avg absolute diff on y\_axis: 0.6496156217923061

Applying avg absolute diff on z\_axis: 0.6049543094942061

Applying min on x\_axis: -1.363360857142857

Applying min on y\_axis: -1.3331605714285715

Applying min on z\_axis: -1.3638615238095237

Applying max on x\_axis: 1.360968380952381

Applying max on y\_axis: 1.3764914000000001

Applying max on z\_axis: 1.33198819047619

Applying max-min diff on x\_axis: 2.7243292380952377

Applying max-min diff on y\_axis: 2.7096519714285714

Applying max-min diff on z\_axis: 2.695849714285714

Applying median on x\_axis: -0.02353086190476192

Applying median on y\_axis: 0.03759409047619046

Applying median on z\_axis: 0.09206555714285715

Applying median abs dev on x\_axis: 0.4542284999999999

Applying median abs dev on y\_axis: 0.6274349095238094

Applying median abs dev on z\_axis: 0.548894542857143

Applying IQR on x\_axis: 0.912718819047619

Applying IQR on y\_axis: 1.2576291285714287

Applying IQR on z\_axis: 1.158704673809524

Applying negative count on x\_axis: 632

Applying negative count on y\_axis: 673

Applying negative count on z\_axis: 716

Applying positive count on x\_axis: 688

Applying positive count on y\_axis: 647

Applying positive count on z\_axis: 604

Applying values above mean on x\_axis: 632

Applying values above mean on y\_axis: 673

Applying values above mean on z\_axis: 716

Applying values below mean on x\_axis: 688

Applying values below mean on y\_axis: 647

Applying values below mean on z\_axis: 604

Applying number of peaks on x\_axis: 451

Applying number of peaks on y\_axis: 442

Applying number of peaks on z\_axis: 441

Applying skewness on x\_axis: 0.09000602953872863

Applying skewness on y\_axis: -0.07910393643284531

Applying skewness on z\_axis: -0.18151097976780978

Applying kurtosis on x\_axis: -0.7231033556102662

Applying kurtosis on y\_axis: -1.181820388478901

Applying kurtosis on z\_axis: -1.0416019685057663

Applying energy on x\_axis: 5.642042413121732

Applying energy on y\_axis: 7.416825685449181

Applying energy on z\_axis: 6.632561512771293

Average resultant acc is 44.3750263226313

Signal magnitude area is 1.786367140079365

De pe urma acestor metrici, valorile obtinute nu indica vreo anomalie evidenta.

**Antrenare de modele ML**

Folosind percentile=10 (valoarea default din sklearn) am fi folosit doar un atribut din cele 9, ceea ce este destul de riscant intrucat se pierde foarte multa informatie pentru fiecare exemplu.

**Rezultate pentru folosirea Select Percentile cu percentile=50, adica folosirea a 4 din 9 atribute per fiecare intrare din dataset**

Text

Description automatically generated

**Rezultate pentru folosirea Select Percentile cu percentile=100, adica folosirea tuturor celor 9 din 9 atribute per fiecare intrare din dataset**

Text, letter

Description automatically generated

Pentru XGBoost learning\_rate joaca un rol foarte important. Pentru SVC, kernelul “rbf” pare a fi cel mai constant dpdv al performantei obtinute. De asemenea, pare ca valoarea de 0.15 pentru “C” este ideala intrucat ofera cele mai bune performante. Random Forest pare ca prefera un numar finit de estimatori si o adancime maxima care nu este infinita. Dar aici intervine si dimensiunea relativ scazuta a setului de date.

Observam ca in mod constant Support Vector Machine Classifier obtine cea mai buna acuratete pe setul de antrenare. Am considerat in continuare modelul antrenat folosind percentile=50, intrucat acuratetea la antrenare este aceeasi, dar volumul de date este considerabil mai mic, imbunatatind astfel performanta.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **General Accuracy** | **Classes** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **Classifiers and Parameters**  *(best performing)* |  |  |  |  |  |  |  |  |  |  |
| **Random Forest**  *Bootstrap: False*  *Max\_depth: 50,*  *N\_estimators: 50* | Train: 0.8  Test: 0.72 | Precision  Recall  F1 | 0.73  0.82  0.77 | 0.66  0.77  0.76 | 0.67  0.87  0.76 | **0.91**  0.52  0.66 | 0.68  0.9  0.77 | 0.81  0.65  0.72 | 0.65  0.52  0.58 | 0.72  0.65  0.68 |
| **Support Vector Machine**  *C: 0.15*  *Kernel: rbf* | Train**: 0.83**  Test: **0.81** | Precision  Recall  F1 | 0.85  0.87  0.86 | 0.88  **0.92**  **0.9** | 0.84  0.9  0.87 | 0.87  0.7  0.77 | 0.69  0.95  0.8 | 0.83  0.72  0.77 | 0.9  0.5  0.65 | 0.72  0.9  0.8 |
| **XGBoost**  *Learning\_rate: 0.2*  *Max\_depth: 2*  *N\_estimators: 150* | Train: 0.73  Test: 0.72 | Precision  Recall  F1 | 0.79  0.77  0.78 | 0.72  0.9  0.8 | 0.74  0.87  0.8 | 0.86  0.62  0.72 | 0.7  0.85  0.77 | 0.79  0.75  0.77 | 0.56  0.35  0.43 | 0.61  0.67  0.64 |

Chart, treemap chart

Description automatically generatedChart

Description automatically generatedChart, treemap chart

Description automatically generated

Mai sus sunt prezentate rezultatele pentru cea mai buna combinatie de parametrii pentru fiecare algoritm, urmarind: acuratetea generala, recall, precision si F1 (ultimele trei la nivel de clasa).

De asemenea, sunt afisate matricile de confuzie pentru acesti algoritmi.

Toate aceste date sunt obtinute de pe urma predictiilor pe setul de testare.

**Etapa 2**

**Analiza datelor pentru PEMS-SF**

Am realizat analiza asupra secventelor de timp pentru 10 cei mai "sensibili" senzori (adica cu variatia standard cea mai mare). Am obtinut urmatoarele rezultate:

Analyzing day number 0.

mean: 0.06314583333333333

std: 0.054040430948349524

min: 0.0

max: 0.329

median: 0.0521

max-min diff: 0.329

IQR: 0.05985

number of peaks: 42

skewness: 1.926261696888133

kurtosis: 5.106245282031292

Analyzing day number 60.

mean: 0.07126944444444444

std: 0.04268916932278181

min: 0.0083

max: 0.156

median: 0.07645

max-min diff: 0.1477

IQR: 0.06155000000000001

number of peaks: 30

skewness: 0.10068100566356297

kurtosis: -1.050568312492597

Analyzing day number 120.

mean: 0.07384791666666667

std: 0.04277140134993825

min: 0.0139

max: 0.2958

median: 0.06820000000000001

max-min diff: 0.2819

IQR: 0.055900000000000005

number of peaks: 36

skewness: 1.3309766490412855

kurtosis: 3.9871911679885024

Analyzing day number 180.

mean: 0.04743819444444444

std: 0.024580166452875786

min: 0.0077

max: 0.092

median: 0.048350000000000004

max-min diff: 0.0843

IQR: 0.04625

number of peaks: 44

skewness: -0.03183009884987836

kurtosis: -1.3613317480516258

Analyzing day number 260.

mean: 0.07223055555555555

std: 0.0573946005950833

min: 0.0099

max: 0.2707

median: 0.0644

max-min diff: 0.2608

IQR: 0.04915

number of peaks: 44

skewness: 1.7583734562826396

kurtosis: 3.062884266543046

Analyzing day number 310.

mean: 0.06564722222222223

std: 0.03377642164218432

min: 0.0203

max: 0.1588

median: 0.0613

max-min diff: 0.1385

IQR: 0.042300000000000004

number of peaks: 47

skewness: 0.7573157586425023

kurtosis: -0.38260250425278297

Analyzing day number 380.

mean: 0.07126944444444444

std: 0.04268916932278181

min: 0.0083

max: 0.156

median: 0.07645

max-min diff: 0.1477

IQR: 0.06155000000000001

number of peaks: 30

skewness: 0.10068100566356297

kurtosis: -1.050568312492597

Analyzing day number 439.

mean: 0.07158958333333335

std: 0.023530951008872217

min: 0.0286

max: 0.119

median: 0.07769999999999999

max-min diff: 0.0904

IQR: 0.041025000000000006

number of peaks: 41

skewness: -0.2603369201727651

kurtosis: -1.270940391218219

**Multi-Layered Perceptron**

Arhitectura folosita este descrisa prin:

- optimizator: Adam

- loss: categorical\_crossentropy

- batch\_size: 16

Straturile folosite sunt:

classifier = Sequential([

Flatten(),

Dropout(0.5),

Dense(256, activation='sigmoid'),

Dropout(0.5),

Dense(128, activation='sigmoid'),

Dropout(0.5),

Dense(8, activation='softmax'),

])

Pe setul de antrenare am obtinut o acuratete de 97%, iar pe cel de test o acuratete de 90%.

**Chart, line chart

Description automatically generated**

**A picture containing graphical user interface

Description automatically generated**

**Retea Neurala Convolutionala**

Arhitectura pe care am implementat-o se foloseste de cele 2 tutoriale indicate in cerinta temei. Foloseste:

- optimizator: Adam

- loss: categorical crossentropy

- batch\_size: 16

Straturile retelei sunt:

classifier = Sequential([

tf.keras.layers.Reshape((315, 3), input\_shape=(315, 3)),

Conv1D(filters=256, kernel\_size=5, activation='relu', padding='same', input\_shape=(315, 3)),

Conv1D(filters=512, kernel\_size=5, activation='relu'),

tf.keras.layers.GlobalAvgPool1D(),

Dense(1024, activation='relu'),

Dense(256, activation='relu'),

Dense(8, activation='softmax')

])

Pe datele de antrenare am obtinut o acuratete de 95%, iar pe datele de test am obtinut o acuratete de 76%.

**Chart, line chart

Description automatically generated**

**A picture containing graphical user interface

Description automatically generated**

**Retea Neurala Recurenta**

Pentru reteaua neurala recurente am experimentat cu mai multi hiperparametrii, dar combinatia care a parut cea mai constanta in rezultate, fara alte tehnici de imbunatatire, a fost urmatoarea:

5classifier = Sequential()

classifier.add(LSTM(256, input\_shape=(315, 3)))

classifier.add(Dense(64, activation='relu'))

classifier.add(Dense(8, activation='softmax'))

classifier.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

Pe setul de antrenare am obtinut o acuratete de 76%, iar pe cel de testare o acuratete de 63%.

Chart, line chart

Description automatically generated

A picture containing application

Description automatically generated

*Multi-Layered Perceptron*

**Train acc: 0.97**

**Test acc: 0.9**

Precision: [**1.**, 0.9375, 0.92857143, 0.86666667, 0.7, 0.84615385, 1., 1.]

Recall: [**1.**, 1., 0.86666667, 0.86666667, 0.93333333, 0.73333333, 1., 0.8]

F1: [**1.**, 0.96774194, 0.89655172, 0.86666667, 0.8, 0.78571429, 1., 0.88888889]

*CNN*

Train acc: 0.95

Test acc: 0.76

Precision: [0.73333333, 0.82352941, 0.75, 0.76923077, 0.71428571, 0.64285714, 0.8125, 0.93333333]

Recall: [0.73333333, 0.93333333, 0.8, 0.66666667, 0.66666667, 0.6, 0.86666667, 0.93333333]

F1: [0.73333333, 0.875, 0.77419355, 0.71428571, 0.68965517, 0.62068966, 0.83870968, 0.93333333]

*RNN*

Train acc: 0.76

Test acc: 0.63

Precision: [0.52631579, 0.66666667, 0.59090909, 0.85714286, 0.78947368, 0.33333333, 0.66666667, 0.5625]

Recall: [0.66666667, 0.8, 0.86666667, 0.8, 1., 0.06666667, 0.4, 0.6]

F1: [0.58823529, 0.72727273, 0.7027027, 0.82758621, 0.88235294, 0.11111111, 0.5, 0.58064516]