Temă Învățare Automată

0. Setup

Instalarea bibliotecilor necesare

```
# !pip install sktime

import numpy as np
import pandas as pd
import seaborn as sns
import sktime
import warnings
import xgboost as xgb
from matplotlib import pyplot as plt
from scipy.signal import argrelextrema, find_peaks
from scipy.stats import kurtosis, skew
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import chi2
from sklearn.feature_selection import VarianceThreshold, SelectPercentile
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sktime.datasets import load_from_tsfile_to_dataframe
from IPython.display import Image
```

Încărcarea seturilor de date

```
traffic_train_x, traffic_train_y = load_from_tsfile_to_dataframe(path + "PEMS-SF_TRAIN.ts")
traffic_test_x, traffic_test_y = load_from_tsfile_to_dataframe(path + "PEMS-SF_TEST.ts")
```

```
traffic_x = pd.concat([traffic_train_x, traffic_test_x], axis=0).reset_index(drop=True)
traffic_y = np.concatenate([traffic_train_y, traffic_test_y])
```

Este reținută și o structura cu toate datele din setul PEMS-SF, pentru a putea aplica graficele de exploatare a datelor pe întreg dataset-ul.

```
gestures_train_x, gestures_train_y = load_from_tsfile_to_dataframe(path + "UWaveGestureLibrary_TRAIN.ts")
gestures_test_x, gestures_test_y = load_from_tsfile_to_dataframe(path + "UWaveGestureLibrary_TEST.ts")

gestures_train_x.columns = ["x", "y", "z"]
gestures_test_x.columns = ["x", "y", "z"]
gestures_y_labels = ["1.0", "2.0", "3.0", "4.0", "5.0", "6.0", "7.0", "8.0"]

gestures_train = gestures_train_x.copy(deep=True)
gestures_train["label"] = gestures_train_y
```

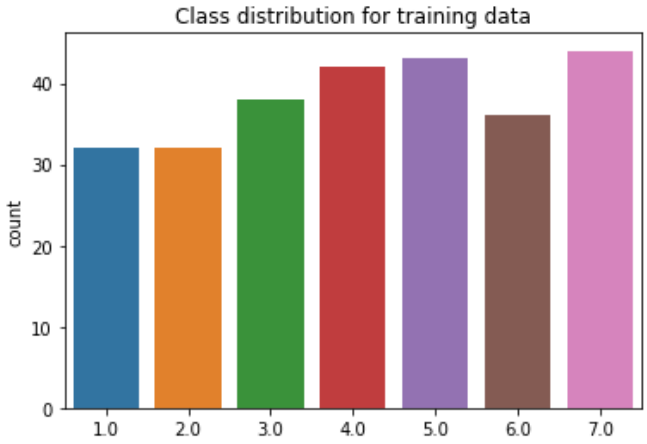
```
# Convert to long-form dataframe. index level=0 is past index (index time series) and index level=1 is time
gestures_train_temp = [[], [], [], [], [], []] # series_index, time, x, y, z, label
IDX, T, X, Y, Z, LABEL = 0, 1, 2, 3, 4, 5

for index, row in gestures_train_x.iterrows():
  # add 315 new values
  series_size = len(row["x"])
  gestures_train_temp[IDX] += [index] * series_size
  gestures_train_temp[T] += list(range(series_size))
  gestures_train_temp[X] += row["x"].tolist()
  gestures_train_temp[Y] += row["y"].tolist()
  gestures_train_temp[Z] += row["z"].tolist()
  gestures_train_temp[LABEL] += [gestures_train_y[index]] * series_size

gestures_lf = pd.DataFrame(np.array(gestures_train_temp).T, columns=["idx", "time", "x", "y", "z", "label"])
```

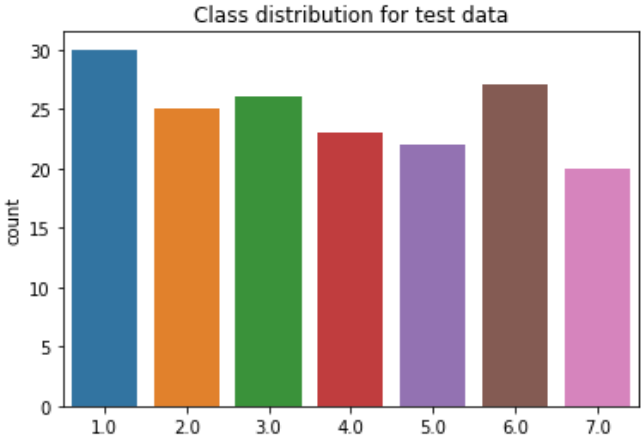
Este reținut și un DataFrame ”long-form”, unde seriile de valori sunt desfășurate pe mai multe linii, astfel:

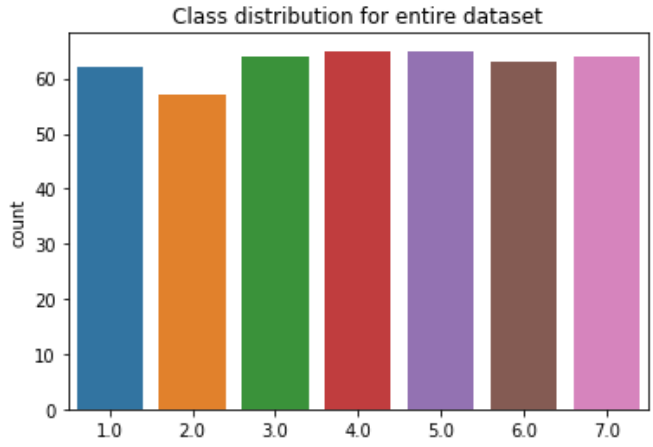
	idx	time	x	y	z	label
0	0	0	0.31745	-1.4345	-0.42612	1.0
1	0	1	0.31745	-1.4345	-0.42612	1.0
2	0	2	0.31745	-1.4345	-0.42612	1.0
3	0	3	0.31745	-1.4345	-0.42612	1.0
4	0	4	0.31745	-1.4345	-0.42612	1.0
...	...	...	...	...	...	...
37795	119	310	0.74769	0.26651	0.21173	8.0
37796	119	311	0.74769	0.24682	0.22527	8.0
37797	119	312	0.74769	0.22788	0.23882	8.0
37798	119	313	0.74769	0.20895	0.25237	8.0
37799	119	314	0.74769	0.19002	0.26592	8.0

1. Exploatarea Datelor

PEMS-SF - Analiza echilibrului claselor



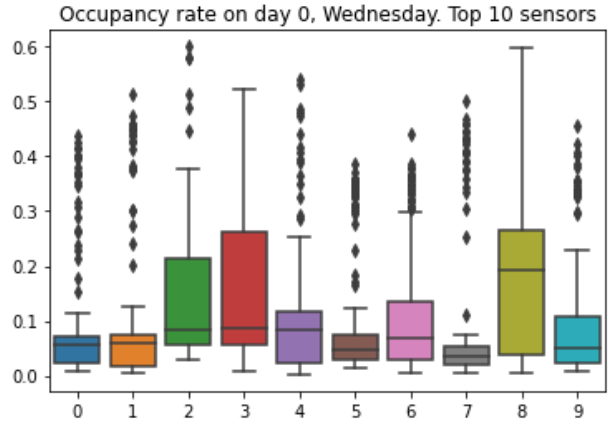


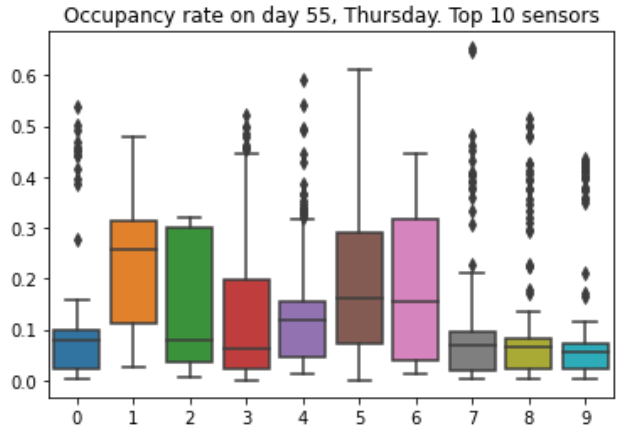


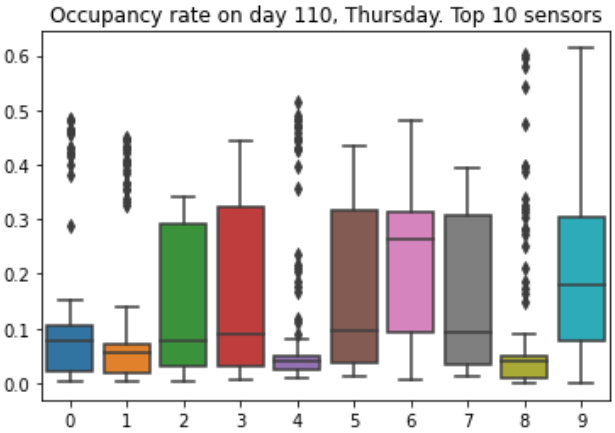
Așa cum ne așteptam, datele sunt echilibrate în ceea ce privește frecvența apariției fiecărei clase. Acest lucru se datorează preluării datelor pe o perioadă contiguă, iar etichetele reprezinta zilele săptămânii.

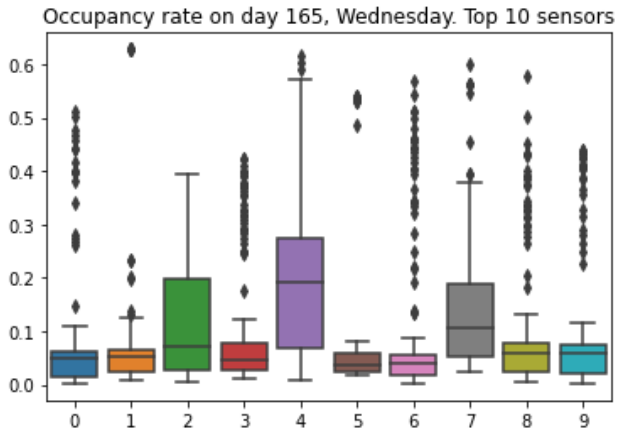
PEMS-SF - Gradul de variere a ratei de ocupare în 8 zile alese uniform

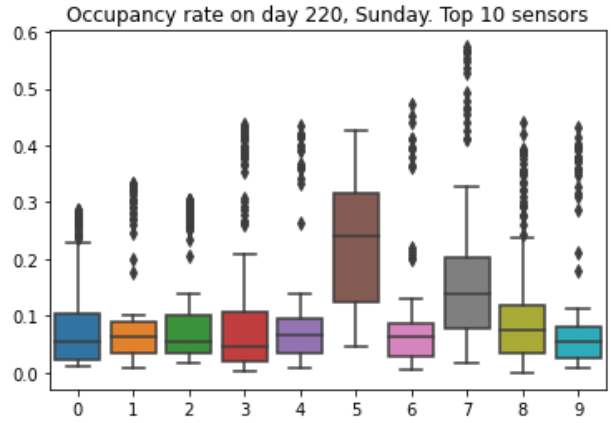
```
weekdays = ["", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

def boxplot_top10sensors_traffic(day):
  row = traffic_x.iloc[day]
  deviations = [sr.std(ddof=0) for sr in row]
  indexes = np.argpartition(deviations, -10)[-10:]
  top10df = pd.DataFrame(np.array([sr.tolist() for sr in row.iloc[indexes]]).T)
  weekday = weekdays[int(float((traffic_y[day])))]
  sns.boxplot(data=top10df).set(title=f"Occupancy rate on day {day}, {weekday}. Top 10 sensors")

nr, nc = traffic_x.shape
for day in np.linspace(0, nr, num=8, endpoint=False, dtype=int):
  boxplot_top10sensors_traffic(day)
  plt.show()
```

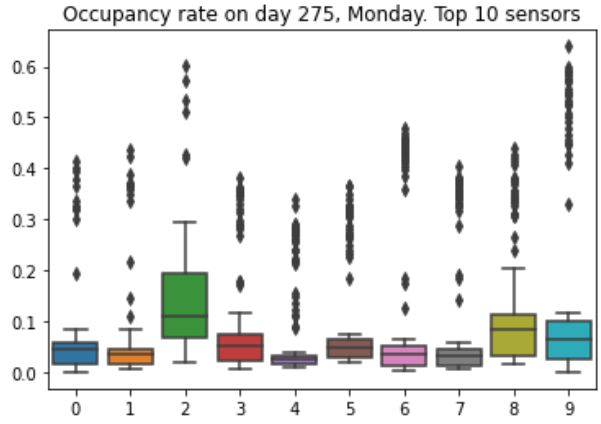

Occupancy rate on day 0, Wednesday. Top 10 sensors

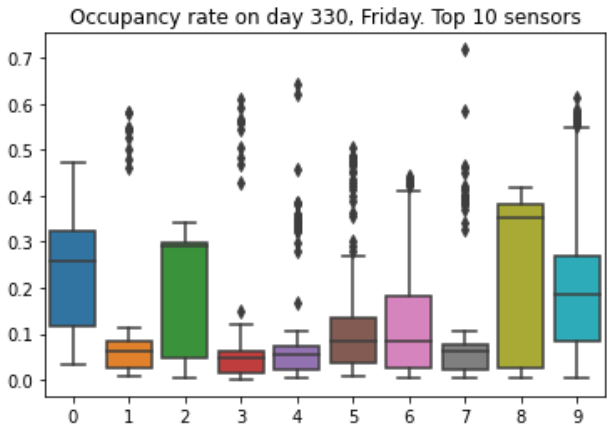

Occupancy rate on day 55, Thursday. Top 10 sensors

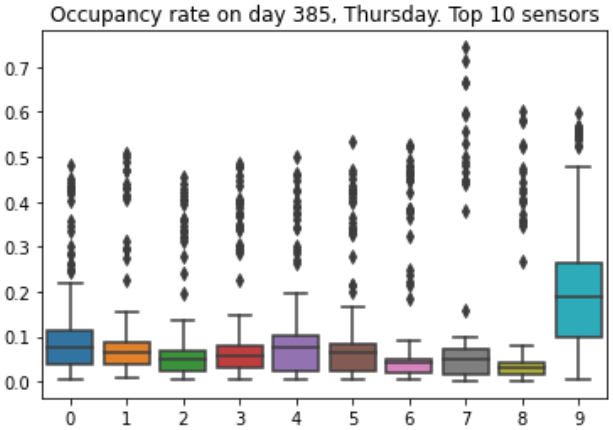

Occupancy rate on day 110, Thursday. Top 10 sensors


Occupancy rate on day 165, Wednesday. Top 10 sensors


Occupancy rate on day 220, Sunday. Top 10 sensors


Occupancy rate on day 275, Monday. Top 10 sensors


Occupancy rate on day 330, Friday. Top 10 sensors


Occupancy rate on day 385, Thursday. Top 10 sensors

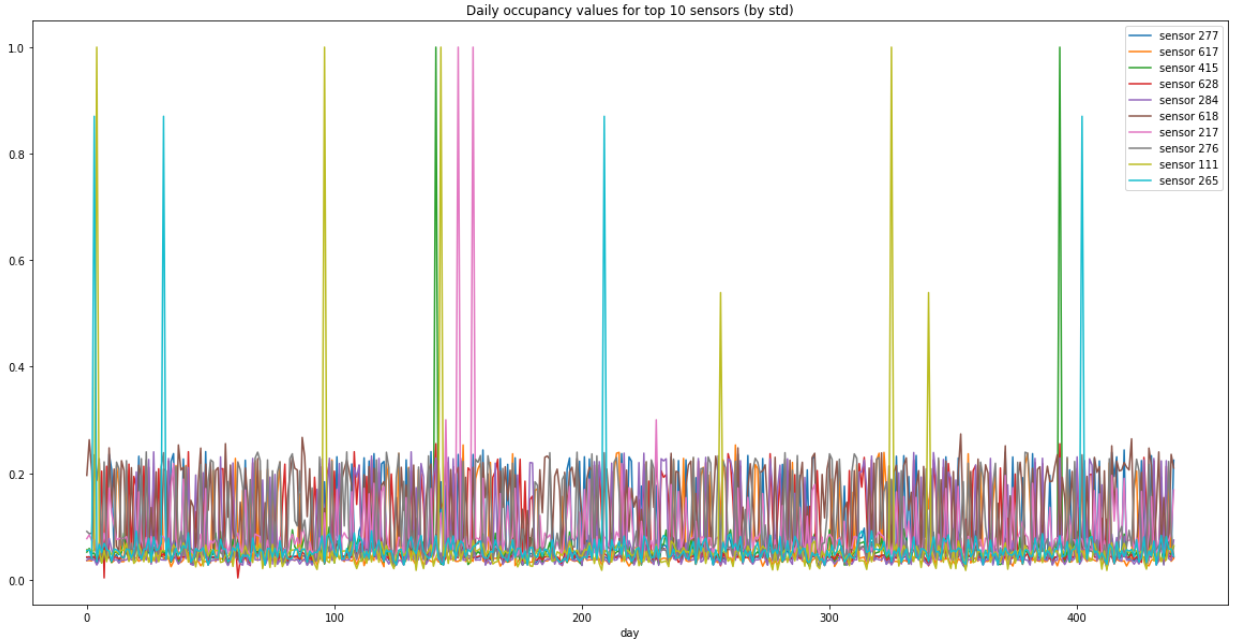
Observații

- Există mulți outliers in valorile înregistrate de senzori într-o zi.
- Există unele zile (ex. ziua 385) când datele sunt strâns grupate și simetrice pentru toți cei 10 senzori. Ziua 385 este o Joi la fel ca zilele 55 și 110, dar datele arata diferit (cel puțin varianța lor). Se poate datora perioadei anului sau existența unei sărbători.
- Pare ca în timpul saptamanii datele sunt mai puțin cuplate și mai nesimetrice decât în weekend, semn ca pot fi mai greu de clasificat zilele marți-vineri.

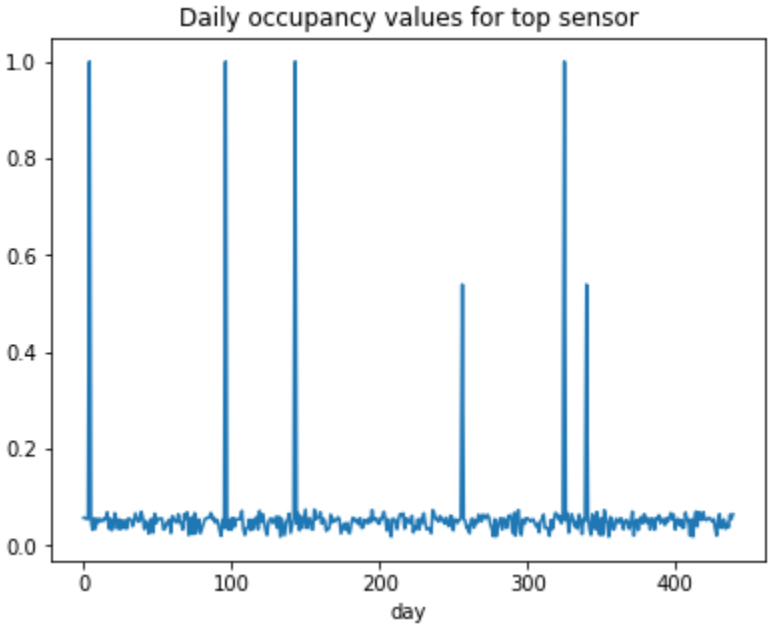
PEMS-SF - Media zilnică a ratei de ocupare pe perioada înregistrată

```
nr, nc = traffic_x.shape
daily_mean_occupancy = [[] for _ in range(nc)]

for day in range(nr):
  row = traffic_x.iloc[day]
  means = [sr.mean() for sr in row]
  for i in range(nc):
    daily_mean_occupancy[i].append(means[i])

# Now we have list of 963 lists containing the mean occupancy values for each day for one sensor.
deviations = [np.std(arr) for arr in daily_mean_occupancy]
indexes = np.argpartition(deviations, -10)[-10:]
```

```
plt.rcParams['figure.figsize'] = [20, 10]

for idx in indexes:
  plt.plot(daily_mean_occupancy[idx], label=f"sensor {idx}")
plt.legend()
plt.title("Daily occupancy values for top 10 sensors (by std)")
plt.xlabel("day")
plt.show()

plt.rcParams['figure.figsize'] = [6.4, 4.8]
```



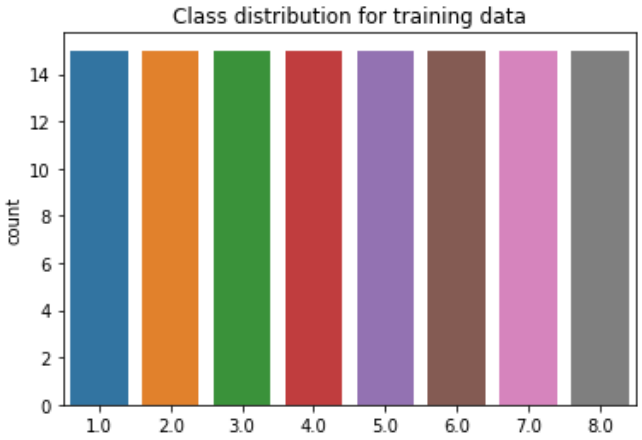
Doar pentru un senzor (cel cu deviația cea mai mare):

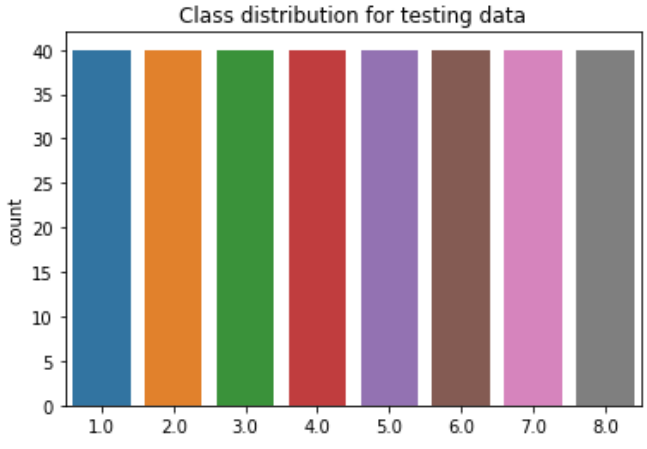


Observații

- Din cei 10 senzori cu varianța cea mai mare, doar 5 se observă în grafic cî "ies" dintr-o zona de valori apropiate (0.1-0.2) și ating ocazional intensități aproape de 1. Acele creșteri bruște pot ajuta la clasificare mai bine decât oscilația constantă a celorlalți 5 senzori.
- În partea de jos a graficului, se observa 2 zone de valori parcurse, de grosimi diferite. Valorile unor senzori oscilează între 0.05 și 0.2, iar o alta gamă de senzori oscilează între 0.05 și 0.1. Deci putem folosi atribute bazate pe minimul și maximul valorilor.

UWaveGestureLibrary - Analiza echilibrului claselor





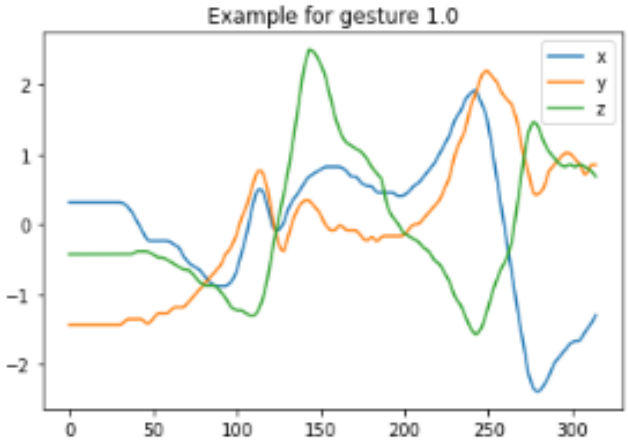
Clasele sunt perfect echilibrate. Înseamnă ca avem un număr egal de exemple pentru fiecare tip de gest.

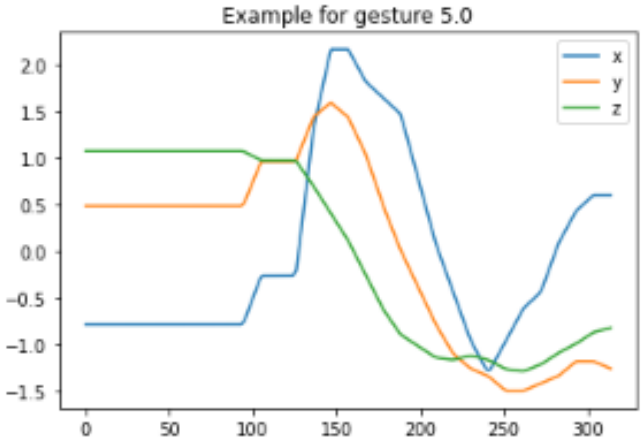
UWaveGesture - Un exemplu de serie pentru fiecare gest

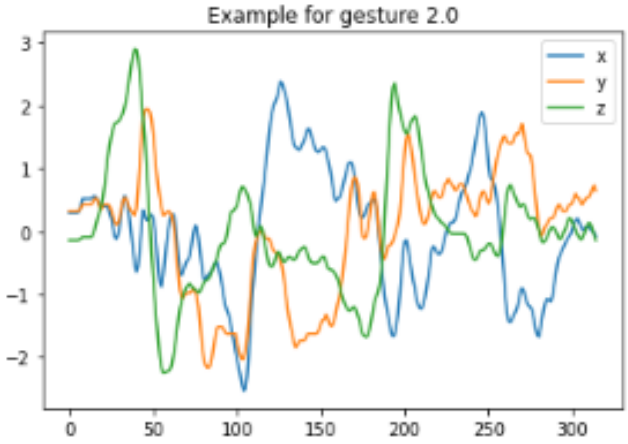
```
def plot_example_for_class(x, y, yi):
  try:
    index = np.where(gestures_train_y==yi)[0][0]
  except Exception:
    print(f"Class {yi} not found in y.")
    return None

  row = x.iloc[index]
  plt.plot(row[0], label="x")
  plt.plot(row[1], label="y")
  plt.plot(row[2], label="z")
  plt.legend()
  plt.title(f"Example for gesture {yi}")
  plt.show()

for label in gestures_y_labels:
  plot_example_for_class(gestures_train_x, gestures_train_y, label)
```

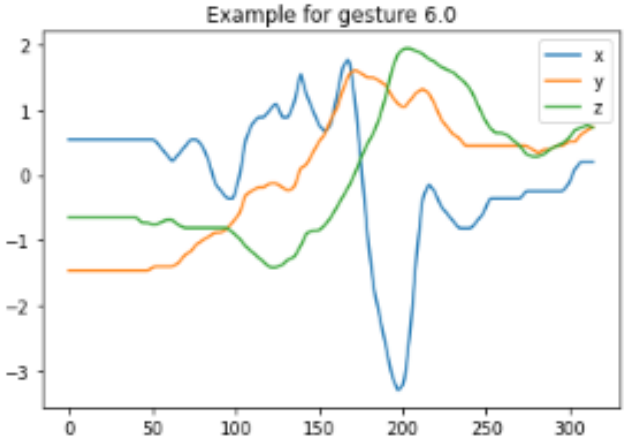
Observații

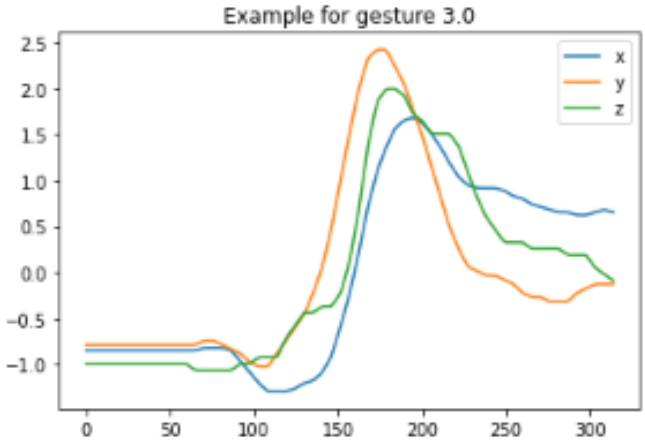
- Se observă asemănări între gesturi.
    - Gesturile 3 si 4 au mișcarea inversă pe x, lucru care se observa și în grafice.
    - Gesturile 5 si 6 au o mișcare asemănătoare, inservata pe axa y.
    - Gesturile 7 si 8 au mișcarea inversă pe x, la fel pe y si z.
- O sa fie mai dificil de clasificat între aceste perechi de gesturi ce pot fi confundate.

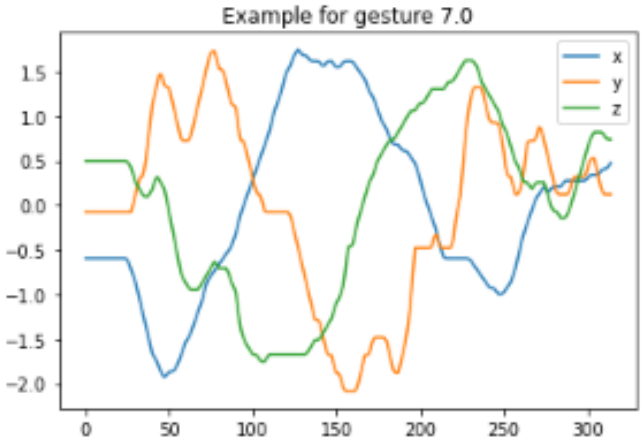

Example for gesture 1.0
x
y
z

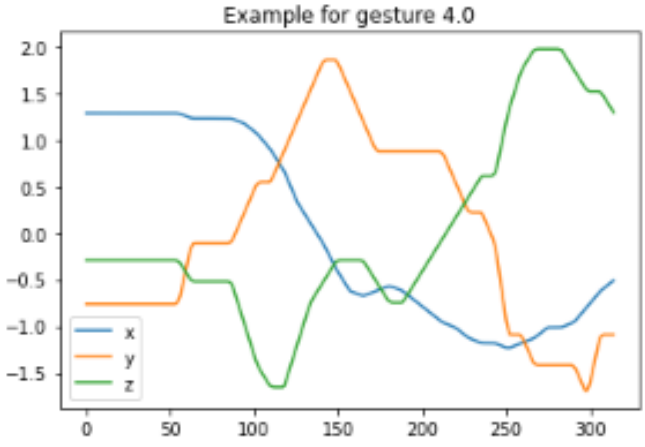

Example for gesture 5.0
x
y
z

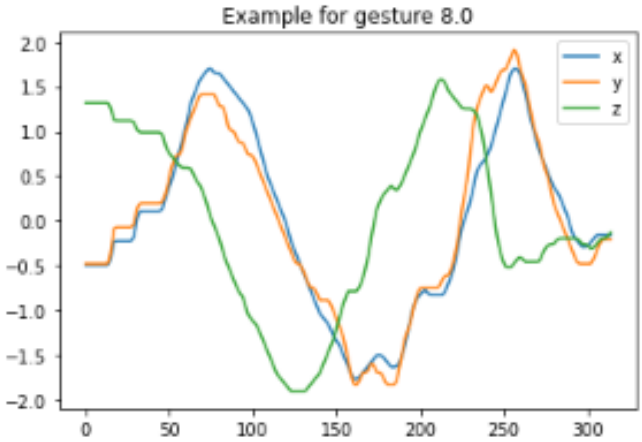

Example for gesture 2.0
x
y
z


Example for gesture 6.0
x
y
z


Example for gesture 3.0
x
y
z

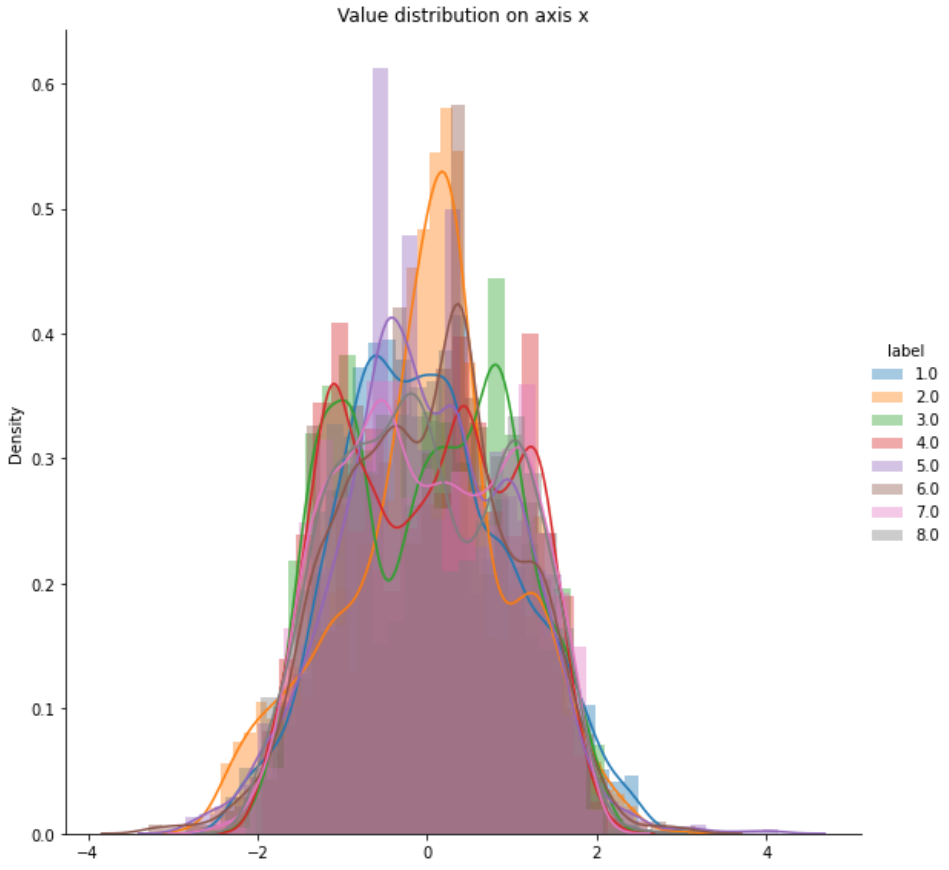

Example for gesture 7.0
x
y
z

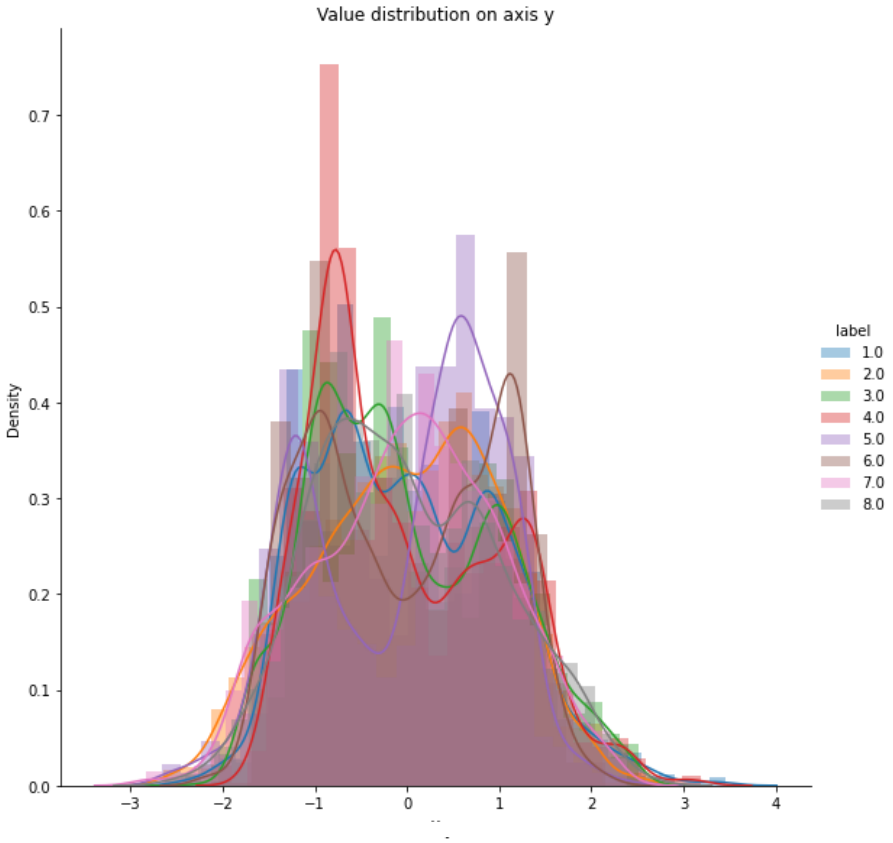

Example for gesture 4.0
x
y
z

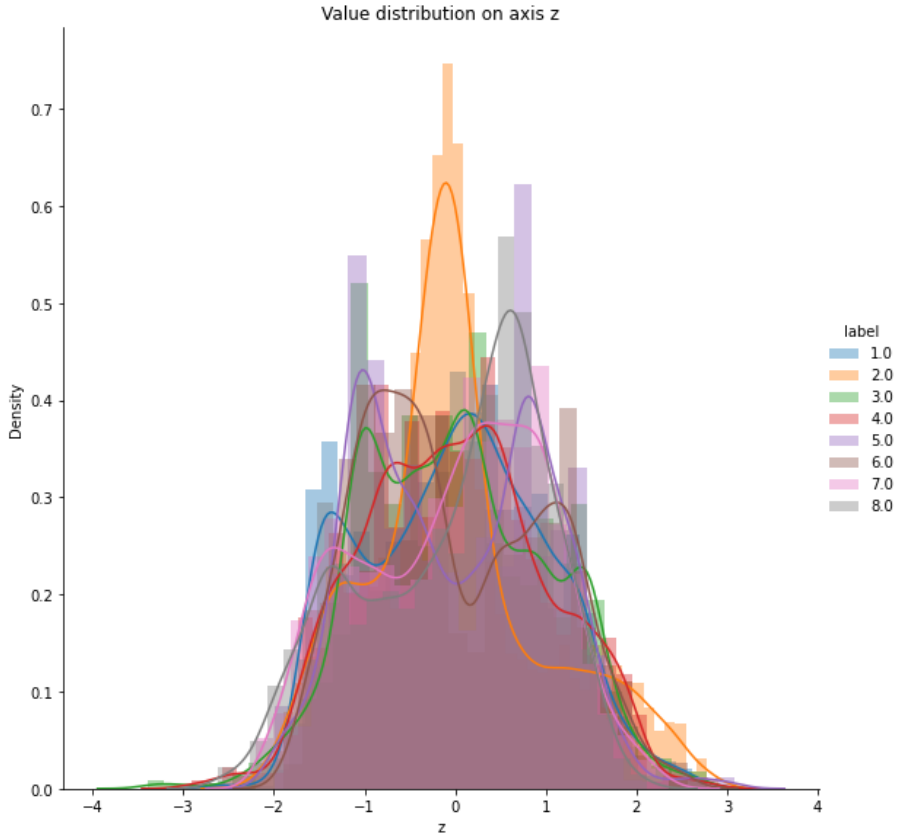

Example for gesture 8.0
x
y
z

UWaveGesture - Distribuția valorilor per fiecară axă per gest

```
sns.FacetGrid(gestures_lf, hue="label", height=8).map(sns.distplot, "x").add_legend().set(title="Value distribution on axis x")
sns.FacetGrid(gestures_lf, hue="label", height=8).map(sns.distplot, "y").add_legend().set(title="Value distribution on axis y")
sns.FacetGrid(gestures_lf, hue="label", height=8).map(sns.distplot, "z").add_legend().set(title="Value distribution on axis z")
```




Value distribution on axis y
Density
label
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0


Value distribution on axis z
Density
z
label
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0

Observații

- Pe axa x, valorile sunt centrare în 0. Densitatea însă diferă.
    - Gestul 2 are valori predominant aproape de 0.
    - Gesturile 7 și 8 au valori ale densitatilor apropiate.
    - Gesturile 3 și 4 au densități simetrice fata de dreapta 0.
    - Gestul 1 are densitati distinctive.
- Pe axa y, valorile ocupă o plajă mai mare. Se observă aceleași pattern-uri legate de perechi de gesturi ce seamănă, dar sunt oglinda.
- Pe axa z, se observă tot aceleași pattern-uri.

2. Extragerea Atributelor și Utilizarea Algoritmilor Clasici de Învățare Automată

PEMS-SF - Extragerea atributelor

```
def extract_features_traffic(row):
  features = np.array([
    sr.agg(['min', 'max', 'std', 'median', 'skew', 'kurt']).tolist() + [
      np.percentile(sr, 0.25), np.percentile(sr, 0.75), len(argrelextrema(np.array(sr), np.greater))
    ]
    for sr in row
  ]).flatten()
  return features
```

```
nr, nc = traffic_train_x.shape
features_arr = [extract_features_traffic(traffic_train_x.iloc[i]) for i in range(nr)]
```

```
nr, nc = traffic_test_x.shape
test_features_arr = [extract_features_traffic(traffic_test_x.iloc[i]) for i in range(nr)]
```

```
trf_train_x = pd.DataFrame(np.array(features_arr))
trf_train_y = traffic_train_y

trf_test_x = pd.DataFrame(np.array(test_features_arr))
trf_test_y = traffic_test_y

cols = np.array([[
  f"min_{i}", f"max_{i}", f"std_{i}", f"median_{i}", f"skew_{i}", f"kurt_{i}",
  f"perc25_{i}", f"perc75_{i}", f"maxno_{i}"
] for i in range(nc)]).flatten()

trf_train_x.columns = trf_test_x.columns = cols
```

PEMS-SF - Selectarea Atributelor

```
def select_features_traffic(train_x, test_x, train_y, threshold=0.1, perc=60):
  # Transform data to [0, 1]
  columns_name = train_x.columns
  scaler = MinMaxScaler()
  train_x = pd.DataFrame(scaler.fit_transform(train_x))
  test_x = pd.DataFrame(scaler.fit_transform(test_x))
  train_x.columns = test_x.columns = columns_name

  # Apply VarianceThreshold selector
  thresholdSel = VarianceThreshold(threshold=threshold)
  thresholdSel = thresholdSel.fit(train_x, train_y)
  cols = thresholdSel.get_support(indices=True).tolist()
  train_x = train_x.iloc[:, cols]
  test_x = test_x.iloc[:, cols]

  # Apply SelectPercentile selector
  percentileSel = SelectPercentile(score_func=chi2, percentile=perc)
  percentileSel = percentileSel.fit(train_x, train_y)
  cols = percentileSel.get_support(indices=True).tolist()
  train_x = train_x.iloc[:, cols]
  test_x = test_x.iloc[:, cols]

  return train_x, test_x
```

```
trf_train_x, trf_test_x = select_features_traffic(trf_train_x, trf_test_x, trf_train_y)
```

Observații

- Numărul total de atribute considerate, rezultate din extragere: 8667.
- Numărul total de atribute selectate pentru a fi folosite în antrenare: 234.
- Se observă mai jos că au fost păstrate atribute care țin de proprietăți variate ale seriilor de numere.

	min_9	perc25_9	perc75_9	skew_10	min_67	min_71	perc25_71	perc75_71	min_81	min_83	...	perc25_931	perc75_931	std_937	min_942	perc25_942	perc75_942	min_950	perc25_950	perc75_950	perc75_960
0	0.093750	0.094750	0.093049	0.088842	0.900383	0.118367	0.128207	0.136545	0.889952	0.025210	...	0.758376	0.796990	0.878743	0.976431	0.971023	0.943914	0.289655	0.280899	0.278596	0.051846
1	0.058594	0.066693	0.076406	0.088244	0.900383	0.106122	0.107269	0.102409	0.779904	0.008403	...	0.703284	0.697466	0.918881	0.454545	0.456668	0.450420	0.241379	0.248206	0.285978	0.041272
2	0.070312	0.074191	0.078375	0.068539	0.865900	0.122449	0.122093	0.112601	0.875598	0.928571	...	0.660256	0.750766	0.831854	0.932660	0.927567	0.901148	0.048276	0.038705	0.041277	0.071776
3	0.667969	0.674917	0.687451	0.046034	0.881226	0.106122	0.115998	0.125463	0.870813	0.000000	...	0.085328	0.121308	0.030147	0.111111	0.087626	0.038937	0.410345	0.399497	0.389904	0.741594
4	0.734375	0.732702	0.730125	0.113529	0.923372	0.097959	0.103494	0.103533	0.889952	0.000000	...	0.101285	0.104902	0.002564	0.084175	0.067501	0.032150	0.420690	0.409983	0.400363	0.620066

PEMS-SF - Utilizarea algoritmului SVM

```
svc_params = {'kernel': ['linear', 'poly', 'rbf'], 'C': [0.1, 1, 10, 100]}
svc = svm.SVC()
svc_grid = GridSearchCV(svc, svc_params)
```

```
svc_grid.fit(trf_train_x, trf_train_y)
```

```
trf_svc_predictions = svc_grid.predict(trf_test_x)
```

```
print(classification_report(trf_test_y, trf_svc_predictions))
```

```
              precision    recall  f1-score   support

         1.0       0.93      0.87      0.90        30
         2.0       0.82      0.72      0.77        25
         3.0       0.57      0.62      0.59        26
         4.0       0.50      0.57      0.53        23
         5.0       0.50      0.45      0.48        22
         6.0       0.88      0.78      0.82        27
         7.0       0.76      0.95      0.84        20

    accuracy                           0.71       173
   macro avg       0.71      0.71      0.70       173
weighted avg       0.72      0.71      0.71       173
```

```
svc_matrix = confusion_matrix(trf_test_y, trf_svc_predictions)
svc_matrix_df = pd.DataFrame(
    svc_matrix,
    index=["true:1", "true:2", "true:3", "true:4", "true:5", "true:6", "true:7"],
    columns=["pred:1", "pred:2", "pred:3", "pred:4", "pred:5", "pred:6", "pred:7"]
  )
svc_matrix_df
```

	pred:1	pred:2	pred:3	pred:4	pred:5	pred:6	pred:7
true:1	26	0	0	0	0	0	4
true:2	1	18	4	2	0	0	0
true:3	0	2	16	5	3	0	0
true:4	0	0	6	13	3	1	0
true:5	0	2	2	6	10	2	0
true:6	0	0	0	0	4	21	2
true:7	1	0	0	0	0	0	19

Matricea de confuzie

- Zilele apropiate între ele pot fi confundate (ex. duminică-luni, vineri-sambata).

- Se confunda zilele cu program asemanator (zilele de munca - miercuri poate fi confundată cu marți, joi sau vineri; zilele de weekend - sambată poate fi confundat cu vineri sau duminică).

Influența hiperparametrilor

kernel	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
poly	{'C': 10, 'kernel': 'poly'}	0.703704	0.907407	0.773585	0.830189	0.679245	0.778826	0.083377	1
poly	{'C': 100, 'kernel': 'poly'}	0.703704	0.888889	0.773585	0.849057	0.660377	0.775122	0.085649	2
linear	{'C': 10, 'kernel': 'linear'}	0.703704	0.814815	0.735849	0.830189	0.754717	0.767855	0.047756	3
linear	{'C': 100, 'kernel': 'linear'}	0.703704	0.814815	0.735849	0.830189	0.754717	0.767855	0.047756	3
rbf	{'C': 100, 'kernel': 'rbf'}	0.685185	0.814815	0.792453	0.830189	0.698113	0.764151	0.060540	5
linear	{'C': 1, 'kernel': 'linear'}	0.592593	0.759259	0.716981	0.792453	0.679245	0.708106	0.069262	6
rbf	{'C': 10, 'kernel': 'rbf'}	0.611111	0.740741	0.679245	0.811321	0.622642	0.693012	0.074999	7
poly	{'C': 1, 'kernel': 'poly'}	0.574074	0.685185	0.603774	0.735849	0.547170	0.629210	0.070601	8
linear	{'C': 0.1, 'kernel': 'linear'}	0.518519	0.370370	0.396226	0.509434	0.490566	0.457023	0.061414	9
poly	{'C': 0.1, 'kernel': 'poly'}	0.407407	0.259259	0.339623	0.320755	0.320755	0.329560	0.047453	10

- În cazul parametrului C de regularizare, valorile 0.1 sau 1 au fost prea mici pentru a obține un rezultat bun, nu erau suficiente pentru a preveni overfit.
- În cazul kernelului, au obținut rezultate mai bune cele polinomiale (de grad 1 sau mai mare).

PEMS-SF - Utilizarea algoritmului RandomForest

```
rfc_params = {
    'n_estimators': [100, 200, 500],
    'max_depth': [2, 4, 6, 7, 8],
    'max_samples': [0.5, 0.75, 0.8, 0.9]
}
rfc = RandomForestClassifier()
rfc_grid = GridSearchCV(rfc, rfc_params)
```

```
rfc_grid.fit(trf_train_x, trf_train_y)
```

```
print(classification_report(trf_test_y, trf_rfc_predictions))
```

```
              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00        30
         2.0       0.89      0.96      0.92        25
         3.0       0.90      0.73      0.81        26
         4.0       0.77      0.87      0.82        23
         5.0       0.82      0.82      0.82        22
         6.0       0.93      0.93      0.93        27
         7.0       1.00      1.00      1.00        20

    accuracy                           0.90       173
   macro avg       0.90      0.90      0.90       173
weighted avg       0.90      0.90      0.90       173
```

Matricea de confuzie

	pred:1	pred:2	pred:3	pred:4	pred:5	pred:6	pred:7
true:1	30	0	0	0	0	0	0
true:2	0	24	1	0	0	0	0
true:3	0	3	20	3	0	0	0
true:4	0	0	2	20	1	0	0
true:5	0	0	0	1	18	3	0
true:6	0	0	0	0	2	25	0
true:7	0	0	0	0	0	0	20

- Unele zile nu pot fi confundate (ex. luni, duminica) probabil datorită unor trend-uri al traficului pentru acele zile.
- Zilele din mijlocul saptamanii pot fi confundate între ele (ex. marti-miercuri-joi-vineri), dar mai putin ca la modelul SVM.

Influența hiperparametrilor

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'max_depth': 8, 'max_samples': 0.8, 'n_estima...	0.870370	0.907407	0.924528	0.962264	0.867925	0.906499	0.035296	1
{'max_depth': 8, 'max_samples': 0.75, 'n_estim...	0.870370	0.888889	0.943396	0.962264	0.849057	0.902795	0.043168	2
{'max_depth': 8, 'max_samples': 0.9, 'n_estima...	0.870370	0.888889	0.924528	0.943396	0.886792	0.902795	0.026913	2
{'max_depth': 7, 'max_samples': 0.8, 'n_estima...	0.870370	0.925926	0.943396	0.886792	0.886792	0.902655	0.027375	4
{'max_depth': 8, 'max_samples': 0.9, 'n_estima...	0.870370	0.888889	0.924528	0.943396	0.867925	0.899022	0.030034	5
{'max_depth': 7, 'max_samples': 0.75, 'n_estim...	0.870370	0.888889	0.924528	0.943396	0.867925	0.899022	0.030034	5

- Adâncimea maximă a unui arbore influențează cel mai mult rezultatele. Instanțele cu max_depth cel mai mare au obținut cele mai buna acuratete.
- Ceilalți hiperparametrii variază în cazul unor instanțe cu rezultate similare. Se distinge doar o îmbunătățire pentru număr mare de arbori și pentru un procent >= 0.75 din input folosit la antrenare.

PEMS-SF - Utilizarea algoritmului GradientBoosted Trees

```
gbt_params = {
    'n_estimator': [100, 200, 500],
    'max_depth': [2, 4, 7, 8],
    'learning_rate': [0.1, 0.01, 0.05],
    'subsample': [0.5, 0.75, 0.9]
}
gbt = xgb.XGBClassifier()
gbt_grid = GridSearchCV(gbt, gbt_params)
```

```
gbt_grid.fit(trf_train_x, trf_train_y)
```

```
print(classification_report(trf_test_y, trf_gbt_predictions))
```

```
              precision    recall  f1-score   support

         1.0       1.00      0.83      0.91        30
         2.0       0.77      0.96      0.86        25
         3.0       0.95      0.77      0.85        26
         4.0       0.59      0.57      0.58        23
         5.0       0.68      0.77      0.72        22
         6.0       1.00      0.93      0.96        27
         7.0       0.83      1.00      0.91        20

    accuracy                           0.83       173
   macro avg       0.83      0.83      0.83       173
weighted avg       0.85      0.83      0.83       173
```

Matricea de confuzie

	pred:1	pred:2	pred:3	pred:4	pred:5	pred:6	pred:7
true:1	25	4	0	0	0	0	1
true:2	0	24	0	1	0	0	0
true:3	0	0	20	5	1	0	0
true:4	0	2	1	13	7	0	0
true:5	0	1	0	3	17	0	1
true:6	0	0	0	0	0	25	2
true:7	0	0	0	0	0	0	20

- Unele zile nu pot fi confundate (ex. luni, marti) probabil datorită unor trend-uri al traficului pentru acele zile.
- Exista confuzii între clase pentru zile apropiate, zile din mijlocul saptamanii sau weekend.

Influența hiperparametrilor

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.907407	0.907407	0.886792	0.924528	0.943396	0.913906	0.018985	1
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.907407	0.907407	0.886792	0.924528	0.943396	0.913906	0.018985	1
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.907407	0.907407	0.886792	0.924528	0.943396	0.913906	0.018985	1
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.907407	0.888889	0.924528	0.905660	0.924528	0.910203	0.013365	4
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.907407	0.888889	0.924528	0.905660	0.924528	0.910203	0.013365	4
...	...	...	...	...	...	...	...	...
{'learning_rate': 0.01, 'max_depth': 2, 'n_est...	0.796296	0.833333	0.735849	0.867925	0.849057	0.816492	0.046692	103
{'learning_rate': 0.01, 'max_depth': 2, 'n_est...	0.796296	0.833333	0.735849	0.867925	0.849057	0.816492	0.046692	103
{'learning_rate': 0.01, 'max_depth': 2, 'n_est...	0.814815	0.814815	0.716981	0.830189	0.849057	0.805171	0.045859	106

- Learning rate preferat este 0.1, probabil 0.01 sau 0.05 sunt prea mici pentru a converge în timp.
- Adâncimea maximă a arborilor are valoarea optimă 7. Înseamnă că e nevoie de o adâncime suficient de mare, dar creșterea la 8 nu aduce o îmbunătățire.
- Procentul pentru sample-urile folosite in antrenare trebuie sa fie din gama de valori mai mari (0.9, 0.75).
- Numărul de arbori nu influenteaza performantele. Cel mai probabil valoarea minima (100) este suficientă.

UWaveGestures - Extragerea Atributelor

```
def extract_features_gestures(row):
  features = []
  # create windows
  for idx in range(0, 315 - window + 1, slide):
    for sr in row:
      arr = np.array(sr.tolist()[idx:(idx + window)])
      features += [
        arr.mean(),                                    # mean
        arr.std(),                                     # std dev
        np.mean(np.absolute(arr - np.mean(arr))),      # avg absolute diff
        arr.min(),                                     # min
        arr.max(),                                     # max
        arr.max() - arr.min(),                         # max-min diff
        np.median(arr),                                # median
        np.median(np.absolute(arr - np.median(arr))),  # median abs dev
        np.percentile(arr, 75) - np.percentile(arr, 25),  # interquartile range
        np.sum(arr < 0),                               # negative count
        np.sum(arr > 0),                               # positive count
        np.sum(arr > arr.mean()),                      # values above mean
        len(find_peaks(arr)[0]),                       # number of peaks
        skew(arr),                                     # skewness
        kurtosis(arr),                                 # kurtosis
        np.sum(arr**2)/window                         # energy
      ]
    # add avg resultant mean
    features.append(np.mean((row[0]**2 + row[1]**2 + row[2]**2)**0.5))
    # add signal magnitude area
    features.append(np.sum(abs(row[0])/window) + np.sum(abs(row[1])/window) + np.sum(abs(row[2])/window))
  return features
```

UWaveGestures - Selectarea Atributelor

```
def select_features_gestures(train_x, test_x, train_y, threshold=0.1):
  # Standardize data
  scaler = MinMaxScaler()
  train_x = pd.DataFrame(scaler.fit_transform(train_x))
  test_x = pd.DataFrame(scaler.fit_transform(test_x))

  # Apply VarianceThreshold selector
  thresholdSel = VarianceThreshold(threshold=threshold)
  thresholdSel = thresholdSel.fit(train_x, train_y)
  cols = thresholdSel.get_support(indices=True).tolist()
  train_x = train_x.iloc[:, cols]
  test_x = test_x.iloc[:, cols]

  return train_x, test_x
```

Observații

- Numărul total de atribute considerate, rezultate din extragere: 400.
- Numărul total de atribute selectate pentru antrenare: 48.

UWaveGestures - Utilizarea algoritmului SVM

```
svc_params = {'kernel': ['linear', 'poly', 'rbf'], 'C': [0.1, 1, 10, 100]}
svc = svm.SVC()
svc_grid = GridSearchCV(svc, svc_params)
```

```
svc_grid.fit(gst_train_x, gst_train_y)
```

```
print(classification_report(gst_test_y, gst_svc_predictions))
```

```
              precision    recall  f1-score   support

         1.0       0.80      0.88      0.83        40
         2.0       0.85      0.97      0.91        40
         3.0       0.88      0.90      0.89        40
         4.0       0.96      0.65      0.78        40
         5.0       0.76      0.88      0.81        40
         6.0       0.73      0.80      0.76        40
         7.0       0.97      0.80      0.88        40
         8.0       0.97      0.95      0.96        40

    accuracy                           0.85       320
   macro avg       0.86      0.85      0.85       320
weighted avg       0.86      0.85      0.85       320
```

Matricea de confuzie

	pred:1	pred:2	pred:3	pred:4	pred:5	pred:6	pred:7	pred:8
true:1	35	0	0	0	0	5	0	0
true:2	0	39	0	0	0	0	1	0
true:3	0	0	36	0	4	0	0	0
true:4	1	1	0	26	6	5	0	1
true:5	2	0	1	0	35	2	0	0
true:6	2	2	3	1	0	32	0	0
true:7	2	4	1	0	1	0	32	0
true:8	2	0	0	0	0	0	0	38

- Cum se observa și vizual, gestul 2 este cel mai greu de confundat.
- Gesturi cu asemanari (mai ales pe anumite axe) pot fi confundate.
- Perechi de gesturi care par oglindite sunt clasificate corect si nu sunt confundate, cel mai probabil datorită extragerii de atribute signifiante.

Influența hiperparametrilor

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'C': 1, 'kernel': 'poly'}	1.000000	0.875000	0.916667	0.833333	0.958333	0.916667	0.058926	1
{'C': 10, 'kernel': 'poly'}	1.000000	0.875000	0.916667	0.833333	0.958333	0.916667	0.058926	1
{'C': 100, 'kernel': 'poly'}	1.000000	0.875000	0.916667	0.833333	0.958333	0.916667	0.058926	1
{'C': 1, 'kernel': 'rbf'}	0.958333	0.833333	0.916667	0.916667	0.916667	0.908333	0.040825	4
{'C': 10, 'kernel': 'rbf'}	1.000000	0.875000	0.958333	0.791667	0.916667	0.908333	0.071686	4
{'C': 100, 'kernel': 'rbf'}	1.000000	0.875000	0.958333	0.791667	0.916667	0.908333	0.071686	4
{'C': 0.1, 'kernel': 'poly'}	0.958333	0.833333	0.958333	0.916667	0.875000	0.908333	0.048591	7
{'C': 0.1, 'kernel': 'rbf'}	0.958333	0.750000	0.916667	0.875000	0.875000	0.875000	0.069722	8
{'C': 1, 'kernel': 'linear'}	0.875000	0.833333	0.916667	0.875000	0.833333	0.866667	0.031180	9

- Hiperparametrul C nu influențează mult rezultatele.
- Kernelul polinomial duce la cele mai mari acurateți. Kernelul liniar nu este suficient pentru a separa clasele.

UWaveGestures - Utilizarea algoritmului RandomForest

```
rfc_params = {
    'n_estimators': [100, 200, 500],
    'max_depth': [2, 4, 6, 8, None],
    'max_samples': [0.5, 0.75, 0.8, 0.9]
}
rfc = RandomForestClassifier()
rfc_grid = GridSearchCV(rfc, rfc_params)
```

```
rfc_grid.fit(gst_train_x, gst_train_y)
```

```
print(classification_report(gst_test_y, gst_rfc_predictions))
```

```
              precision    recall  f1-score   support

         1.0       0.78      0.88      0.82        40
         2.0       0.72      0.97      0.83        40
         3.0       0.79      0.93      0.85        40
         4.0       0.96      0.62      0.76        40
         5.0       0.70      0.88      0.78        40
         6.0       0.80      0.60      0.69        40
         7.0       0.96      0.68      0.79        40
         8.0       0.93      0.93      0.93        40

    accuracy                           0.81       320
   macro avg       0.83      0.81      0.81       320
weighted avg       0.83      0.81      0.81       320
```

Matricea de confuzie

	pred:1	pred:2	pred:3	pred:4	pred:5	pred:6	pred:7	pred:8
true:1	34	1	0	0	0	5	0	0
true:2	0	39	0	0	0	0	1	0
true:3	0	0	34	0	4	2	0	0
true:4	0	1	0	25	9	2	0	3
true:5	2	0	1	0	34	3	0	0
true:6	4	0	6	1	0	29	0	0
true:7	1	8	1	0	0	0	30	0
true:8	2	0	0	0	0	0	0	38

- Cum se observa si vizual, gestul 2 este cel mai greu de confundat.
- Cateva confuzii intre gesturi ce arata similar pe una sau 2 axe.

Influența hiperparametrilor

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'max_depth': 6, 'max_samples': 0.9, 'n_estima...	1.000000	0.750000	0.916667	0.875000	0.875000	0.883333	0.080795	1
{'max_depth': None, 'max_samples': 0.9, 'n_est...	1.000000	0.750000	0.916667	0.875000	0.875000	0.883333	0.080795	1
{'max_depth': 6, 'max_samples': 0.75, 'n_estim...	1.000000	0.750000	0.916667	0.833333	0.875000	0.875000	0.083333	3
{'max_depth': None, 'max_samples': 0.8, 'n_est...	1.000000	0.750000	0.875000	0.875000	0.875000	0.875000	0.079057	3
{'max_depth': 8, 'max_samples': 0.9, 'n_estima...	0.958333	0.750000	0.916667	0.875000	0.875000	0.875000	0.069722	3
{'max_depth': 8, 'max_samples': 0.8, 'n_estima...	0.958333	0.750000	0.916667	0.875000	0.875000	0.875000	0.069722	3
{'max_depth': 6, 'max_samples': 0.75, 'n_estim...	1.000000	0.750000	0.875000	0.875000	0.875000	0.875000	0.079057	3

- Scorul variază de la 0.88 la 0.82 indiferent de hiperparametrii.
- Par de preferat: adancime maxima mai mare (sau None, i.e. oricat este nevoie), procent mare folosit la antrenare, numar mare de arbori. Deși sunt obtinute scoruri favorabile și în variațiuni diferite ale hiperparametrilor.

UWaveGestures - Utilizarea algoritmului GradientBoosted Trees

```
gbt_params = {
    'n_estimator': [100, 200, 500],
    'max_depth': [2, 4, 7, 8],
    'learning_rate': [0.1, 0.01, 0.05],
    'subsample': [0.5, 0.75, 0.9]
}
gbt = xgb.XGBClassifier()
gbt_grid = GridSearchCV(gbt, gbt_params)
```

```
gbt_grid.fit(gst_train_x, gst_train_y)
```

```
print(classification_report(gst_test_y, gst_gbt_predictions))
```

```
              precision    recall  f1-score   support

         1.0       0.70      0.82      0.76        40
         2.0       0.84      0.93      0.88        40
         3.0       0.79      0.82      0.80        40
         4.0       0.88      0.57      0.70        40
         5.0       0.74      0.85      0.79        40
         6.0       0.80      0.70      0.75        40
         7.0       0.86      0.62      0.72        40
         8.0       0.71      0.90      0.79        40

    accuracy                           0.78       320
   macro avg       0.79      0.78      0.77       320
weighted avg       0.79      0.78      0.77       320
```

Matricea de confuzie

	pred:1	pred:2	pred:3	pred:4	pred:5	pred:6	pred:7	pred:8
true:1	34	1	0	0	0	5	0	0
true:2	0	39	0	0	0	0	1	0
true:3	0	0	34	0	4	2	0	0
true:4	0	1	0	25	9	2	0	3
true:5	2	0	1	0	34	3	0	0
true:6	4	0	6	1	0	29	0	0
true:7	1	8	1	0	0	0	30	0
true:8	2	0	0	0	0	0	0	38

- Cum se observa și vizual, gestul 2 este cel mai greu de confundat.
- Cateva confuzii între gesturi ce arată similar pe una sau 2 axe.

Influența hiperparametrilor

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'learning_rate': 0.1, 'max_depth': 8, 'n_esti...	0.916667	0.750000	0.791667	0.833333	0.875000	0.833333	0.058926	1
{'learning_rate': 0.1, 'max_depth': 8, 'n_esti...	0.916667	0.750000	0.791667	0.833333	0.875000	0.833333	0.058926	1
{'learning_rate': 0.1, 'max_depth': 8, 'n_esti...	0.916667	0.750000	0.791667	0.833333	0.875000	0.833333	0.058926	1
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.916667	0.750000	0.791667	0.833333	0.875000	0.833333	0.058926	1
{'learning_rate': 0.1, 'max_depth': 7, 'n_esti...	0.916667	0.750000	0.791667	0.833333	0.875000	0.833333	0.058926	1
...	...	...	...	...	...	...	...	...
{'learning_rate': 0.01, 'max_depth': 8, 'n_est...	0.750000	0.708333	0.750000	0.750000	0.833333	0.758333	0.040825	100
{'learning_rate': 0.01, 'max_depth': 7, 'n_est...	0.750000	0.708333	0.750000	0.750000	0.833333	0.758333	0.040825	100
{'learning_rate': 0.01, 'max_depth': 7, 'n_est...	0.750000	0.708333	0.750000	0.750000	0.833333	0.758333	0.040825	100
{'learning_rate': 0.01, 'max_depth': 8, 'n_est...	0.750000	0.708333	0.750000	0.750000	0.833333	0.758333	0.040825	100

- Learning rate optim este 0.1. Celelalte variante (0.01 si 0.05) probabil făceau modelul să conveargă mult prea încet.
- E nevoie de o adâncime a arborilor mare.
- Numărul de arbori nu influențează prea mult scorul.

Rezultate

PEMS-SF - Rezultate

	General				1			2			3			4			5			6			7		
Algoritm	A	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SVC C=10 kernel='poly'	0.71	0.72	0.71	0.71	0.93	0.87	0.90	0.82	0.72	0.77	0.57	0.62	0.59	0.50	0.57	0.53	0.50	0.45	0.48	0.88	0.78	0.82	0.76	0.95	0.84
RandomForest max_depth=7 max_samples=0.9 n_estimators=500	**0.9**	**0.9**	**0.9**	**0.9**	**1**	**1**	**1**	**0.89**	**0.96**	**0.92**	0.9	0.73	0.81	**0.77**	**0.87**	**0.82**	**0.82**	**0.82**	**0.82**	0.93	**0.93**	0.93	**1**	**1**	**1**
GradientBoosted Trees max_depth=7 n_estimators=100 subsample=0.9	0.83	0.85	0.83	0.83	**1**	0.83	0.91	0.77	**0.96**	0.86	**0.95**	**0.77**	**0.85**	0.59	0.57	0.58	0.68	0.77	0.72	**1**	**0.93**	**0.96**	0.83	**1**	0.91

Observații

- Metoda de Random Forest are cele mai bune rezultate. Divizarea exemplelor după atribute este o metodă buna.
- SVC are cele mai proaste rezultate. Datele nu sunt ușor separabile printr-o margine de separare.
- Unele zile se disting mai ușor (cum ar fi zilele de weekend - datorită unui trafic mai mare și zilele de Luni si Marti - posibil datorită unui trafic specific). Zilele de Joi și Vineri au cele mai mici valori la metrici, o posibilă explicație ar fi asemănarea dintre ele cand vine vorba de obiceiurile oamenilor.

UWaveGestures - Rezultate

	General				1			2			3			4			5			6			7			8		
Algoritm	A	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SVC C=1 kernel='poly'	**0.85**	**0.86**	**0.85**	**0.85**	**0.8**	**0.88**	**0.83**	**0.85**	**0.97**	**0.91**	**0.88**	0.9	**0.89**	**0.96**	**0.65**	**0.78**	**0.76**	**0.88**	**0.81**	0.73	**0.8**	**0.88**	**0.97**	**0.8**	**0.88**	**0.97**	**0.95**	**0.96**
RandomForest max_depth=4 max_samples=0.75 n_estimators=200	0.81	0.83	0.81	0.81	0.78	**0.88**	0.82	0.72	**0.97**	0.83	0.79	**0.93**	0.85	**0.96**	0.62	0.76	0.7	**0.88**	0.78	**0.8**	0.6	0.69	0.96	0.68	0.79	0.93	0.93	0.93
GradientBoosted Trees max_depth=4 n_estimators=100 subsample=0.5	0.78	0.79	0.78	0.77	0.7	0.82	0.76	0.84	0.93	0.88	0.79	0.82	0.8	0.88	0.57	0.7	0.74	0.85	0.79	**0.8**	0.7	0.75	0.86	0.62	0.72	0.71	0.9	0.79

Observații

- Metoda SVC are cele mai bune rezultate. Datele pot fi împărțite prin margini de separare.
- Metodele bazate pe arbori de decizie au rezultate suficient de bune, dar usor mai slabe.
- Există câteva gesturi din cele 8 mai greu de recunoscut, cum ar fi gesturile 5 și 6. Vizual, cele 2 gesturi arată la fel, doar cu componenta y inversata. Acest lucru poate indica că ele au fost confundate. Asemănător se poate și pentru 3 și 4.