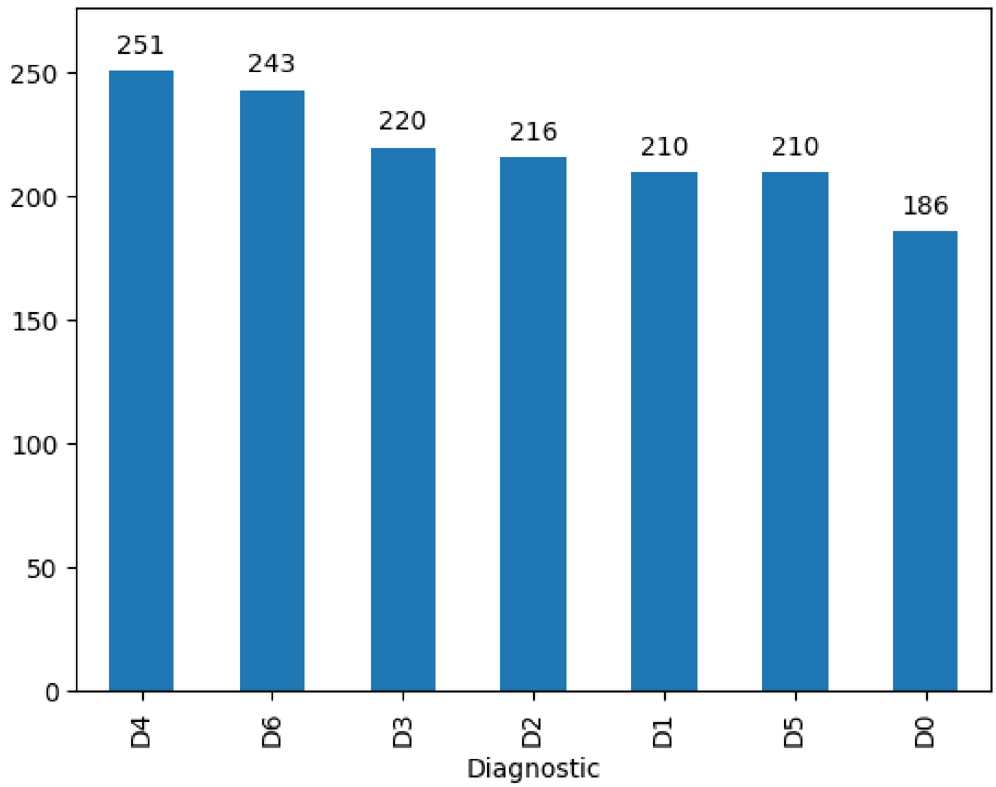
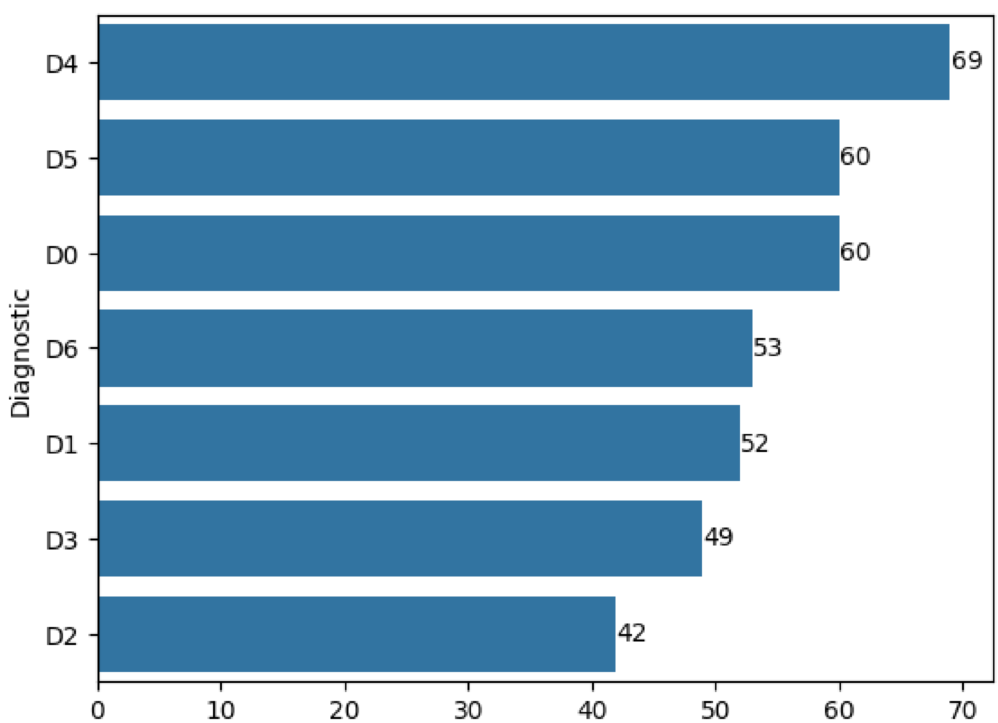
1. Analiza echilibrului de clase

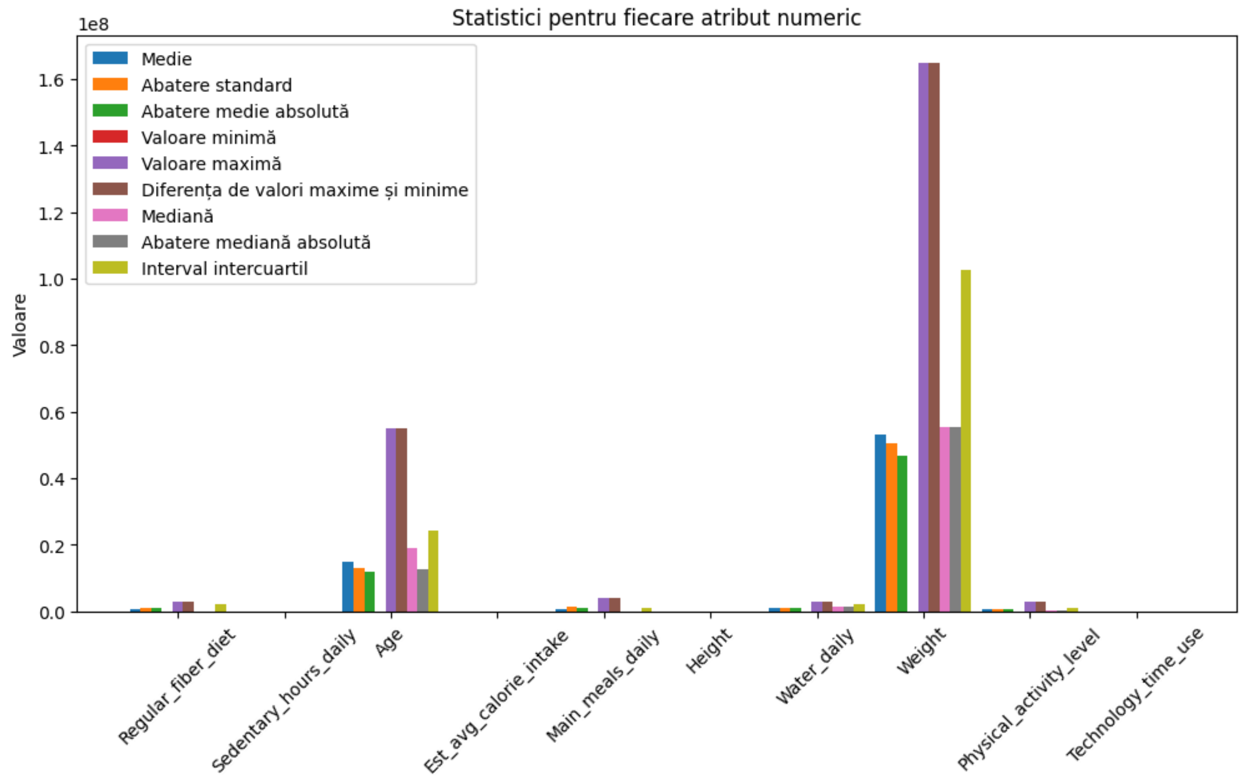
- Frecventa pentru setul de antrenare:



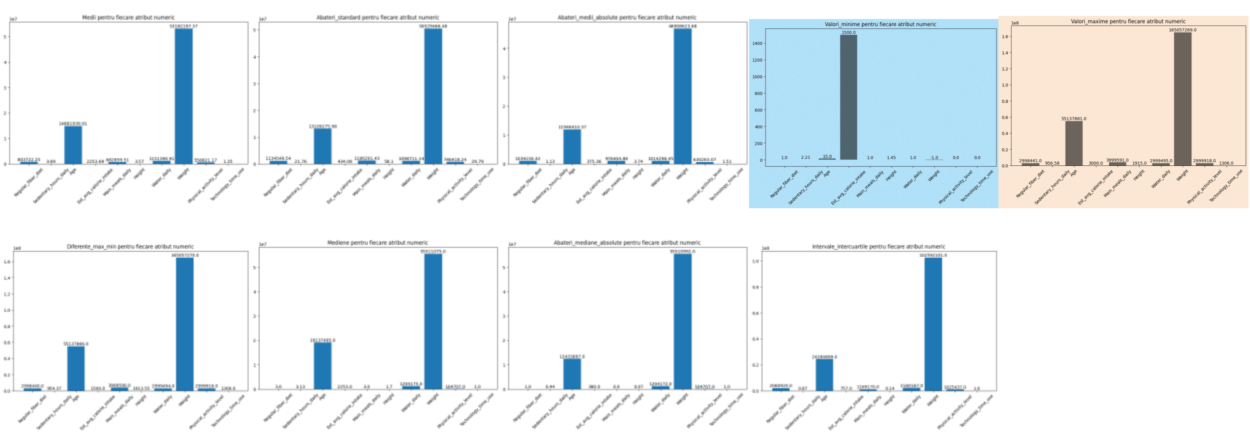
- Frecventa pentru setul de test:



2. Vizualizarea datelor:



Nu prea ne ajuta cu ninmic asa ca luam fiecare statistica in parte:



Interpretare:

Fig.1(Medie) – Se observa multe anomalii, ceea ce indica catre valori eronate. Aici par a fi date normale doar la atributele

“Sedentary_hours_daily”, “Est_avg_calorie_intake” si “Technology_time_use”.

Fig. 4, 5(Valori Minime si Maxime) – Pe Minim singura observatie este atributul “Weight” = -1. Pe Maxim ca si “Sedentary_hours_daily”, si “Technology_time_use” au valori care nu corespund comportamentului uman deci au valori eronate.

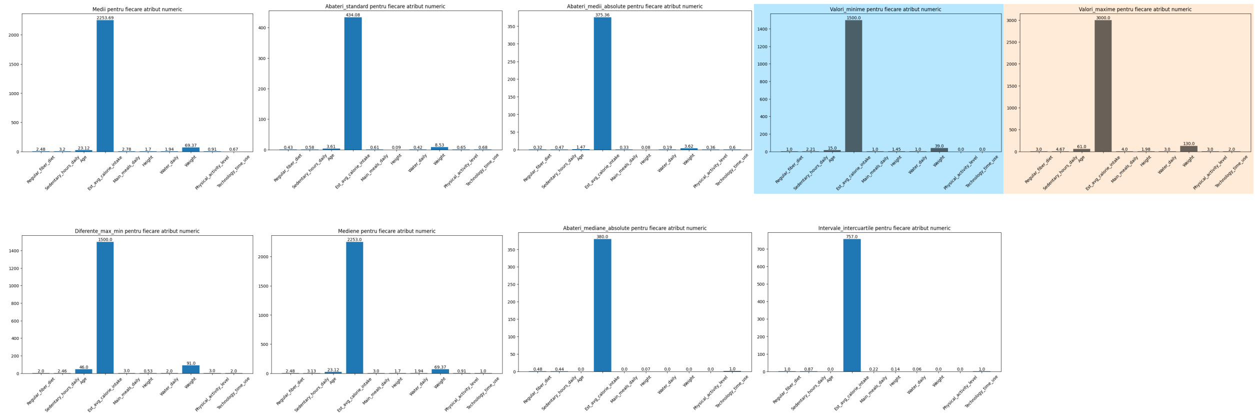
In concluzie din atributele numerice doar “Est_avg_calorie_intake” are valori valide deci el ramane neschimbat.

Imputatie:

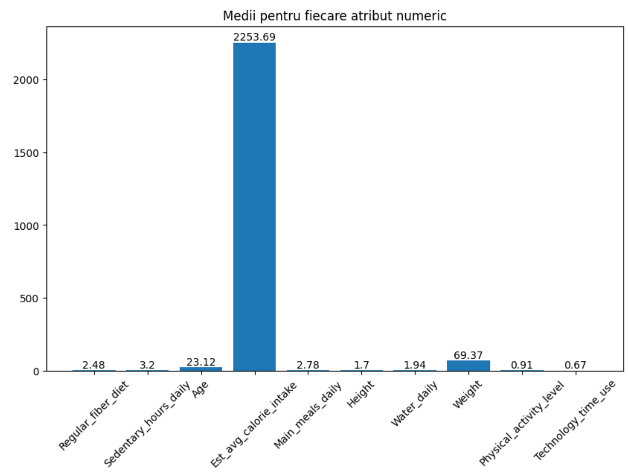
Nu avem probleme decal la EST_avg_calorie_intake deci la celelalte aplicam petoda de Imputatie, cu strategia “mean” dar mai intai trebuie sa inlocuim peste tot unde avem valori eronate cu -1 ca flag. Pentru asta am creat un set de functii care fac clamp intr-un interval anumit bazat pe statistici:

```python
def replace_invalid_diet(diet):
    if diet < 0 or diet > 1000: # o dieta sanatoasa 25-34 statistic oamenii
                                # o incalca asa ca decid sa o fac intre 0-1000
        return -1
    else:
        return diet

def replace_sedentary_hours(hours):
    if hours < 0 or hours > 24:
        return -1
    else:
        return hours

def replace_invalid_age(age):
    if age < 0 or age > 122: # cel mai batran om inregistrat 122 ani
        return -1
    else:
        return age

# Pentru Est_avg_calorie_intake e ok

def replace_main_meals(meals):
    if meals < 0 or meals > 20:
        return -1
    else:
        return meals

def replace_height(heihgt):
    if heihgt < 0.24 or heihgt > 2.72: # 0.24-2.72
        return -1
    else:
        return heihgt

def replace_water_daily(water):
    if water < 0 or water > 10: # (3.7 liters) of fluids a day am pus 10
                                # deoarece pentru sportivi poate fi altfel
        return -1
    else:
        return water

def replace_weight(weight):
    if weight < 0 or weight > 200: # 635kg the heaviest, avem oameni average
                                    # asa ca voi limita la 200
        return -1
    else:
        return weight

def replace_physical_activity_level(level):
    if level < 0 or level > 20: # vad ca in general maxim e in jur de 3 asa ca
                            # fac clamp la 20 in caz de orice inseamna acest level
        return -1
    else:
        return level

def replace_technology_time(hours):
    if hours < 0 or hours > 24:
        return -1
    else:
        return hours
```

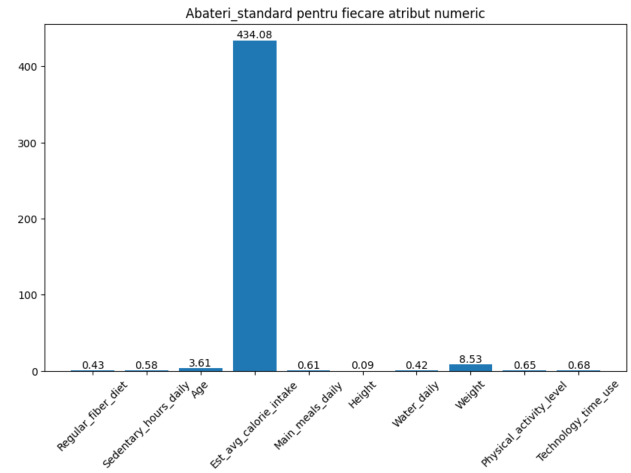
Dupa ce aplicam “SimpleImputer” obtinem urmatoarele statistici:



Interpretare:

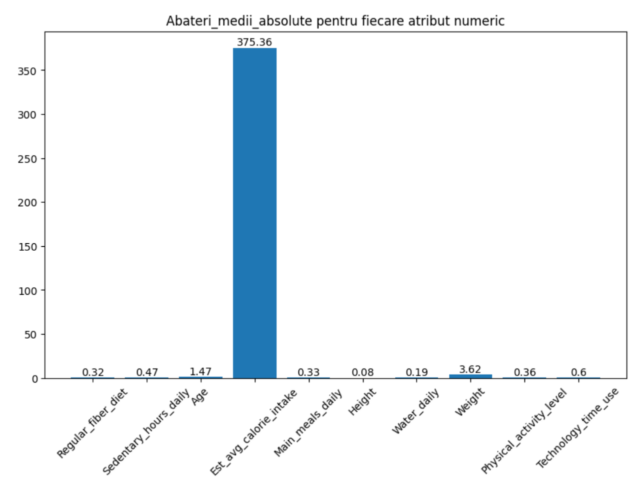


Obtinem indivizi cu parametri “average” in directia favorabila deoarece avem indivizi tineri (23 ani) cu consum de calorii situat in norma recomandata (2000 - 2500), cel din urma corespunde cu un parametru “accurate” in cea ce priveste greutatea (69 kg). Atributul “Regular_fiber_diet” pare usore nefavorabil deoarece 25-34g reprezinta norma sanatoasa insa desi am facut clamp [0, 1000] avem 2.48 valoare medie deci avem indivizi care consuma prea putine fibre, lucru care este acceptabil. “Physical_activity_level” normal este intre 1.4-2.4 deci avem populatie usor sedentara.



O abatere standard mică indică faptul că valorile tind să fie aproape de media (sau valoarea medie) a setului, în timp ce o abatere standard mare indică faptul că valorile sunt răspândite pe o gamă mai largă.

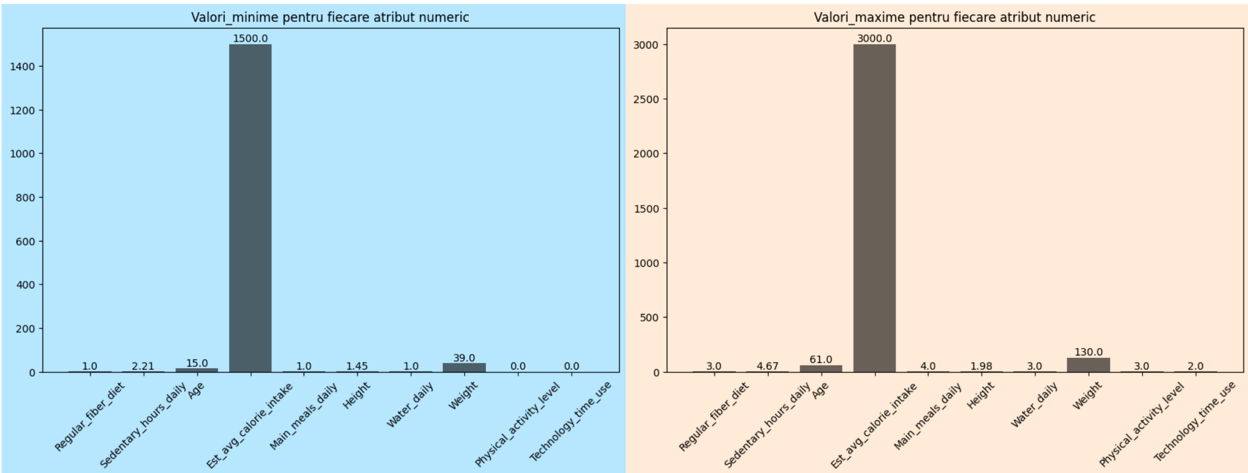
Deci avem o gama larga a atributelor pentru “Est_avg_calorie_intake”, “Weight” si poate "Age"



Observam diferente minore fata de abaterea standard ceea ce indica ca nu avem valori extreme mult mai mari decat medie cee ace indica o distributie echilibrata.

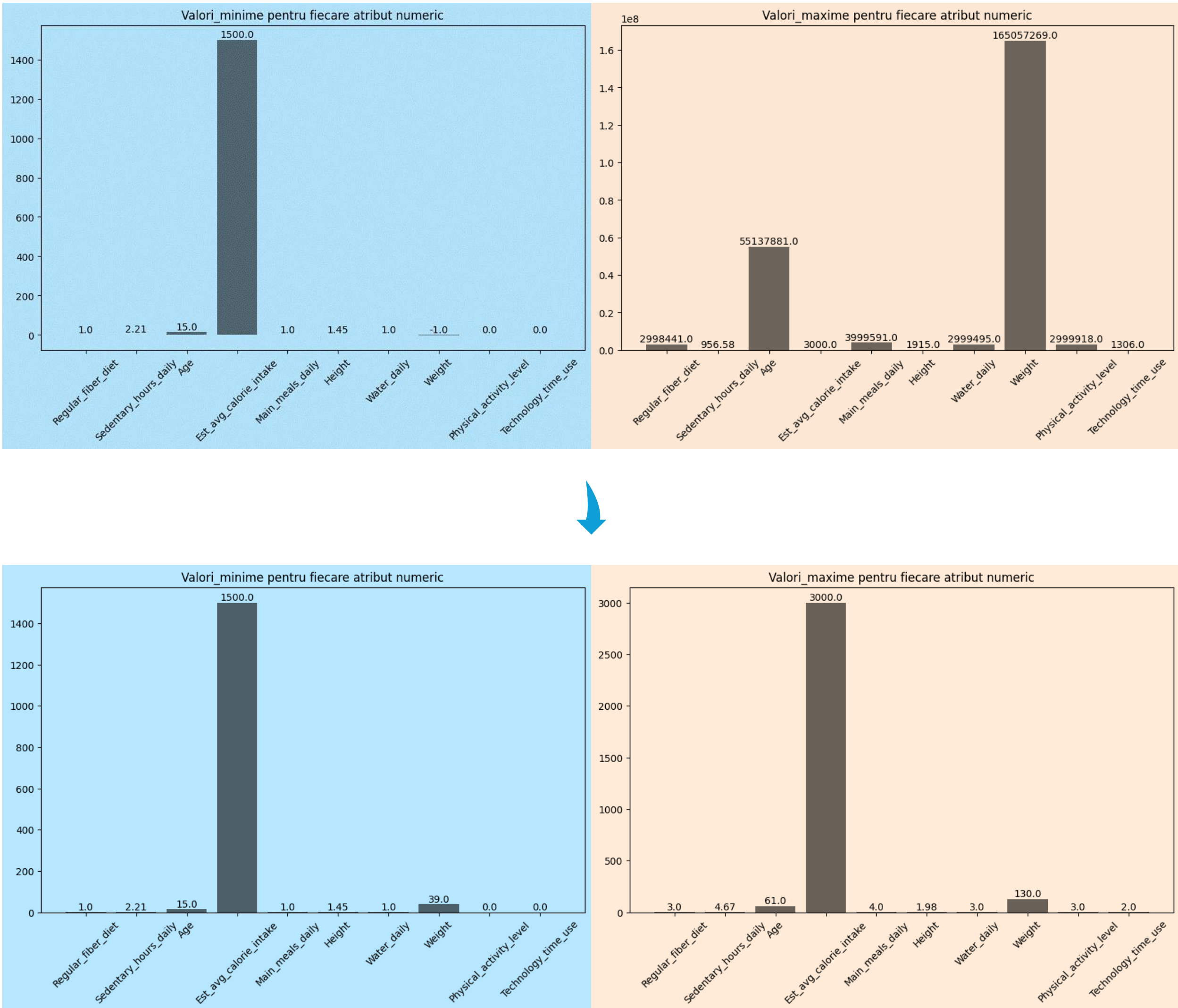
Abaterea Standard – diferentele valoare-medie se ridicau la patrat => obtineam o valoare mult mai mare cee ace ar fi influentat mult abaterea.

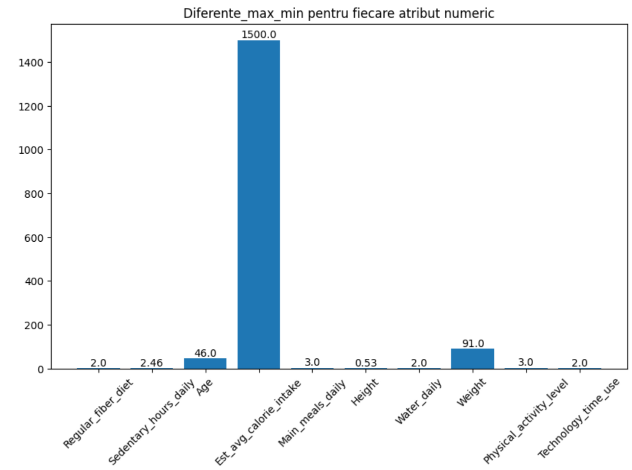
Abaterea Medie – nu ridica la patrat diferentele deci obtinem o evaluare mai precisa in ceea ce priveste distributia valorilor.



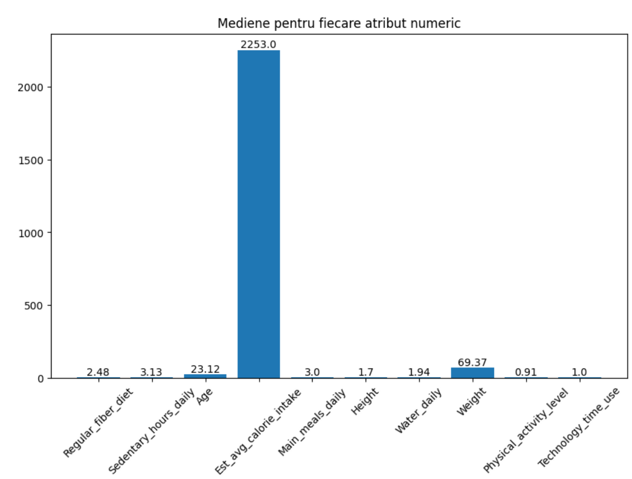
Valorile minime nu sufera modificari decat in cazurile valorilor lipsa(-1) ceea ce este ok. Observam imbunatatiri la valorile maxime, de observant ca am ales intervale de clamp bune deoarece nu avem limite in acele valori => cele care depaseau limitele logice, le depaseau cu mult deci erau clar valori eronate iar cele care se afla in interval nu se afla la pe limita (weight clamp[0 - 200] iar weight max = 130 deci nu am eliminat potentiali 201, 202 care ar putea fi indivizi valizi insa care ar afecta o analiza “average” deoarece ei nu sunt “average”).

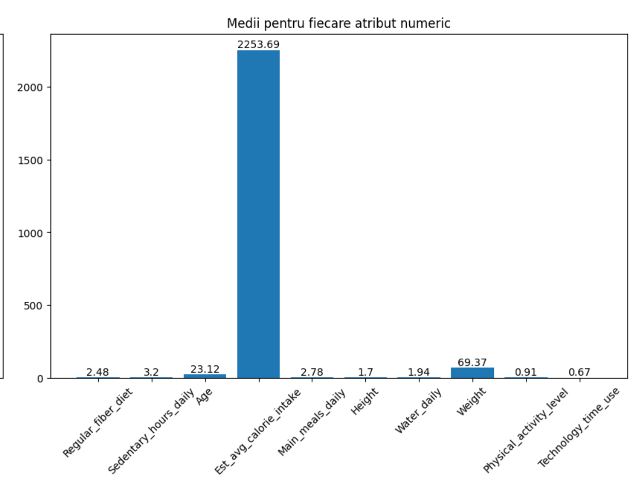
Las mai jos comparatia dintre valorile initiale si cele actuale:


Valori_minime pentru fiecare atribut numeric
1500.0
1.0
2.21
15.0
1.0
1.45
1.0
-1.0
0.0
0.0
Regular_fiber_diet
Sedentary_hours_daily
Age
Est_avg_calorie_intake
Main_meals_daily
Height
Water_daily
Weight
Physical_activity_level
Technology_time_use
Valori_maxime pentru fiecare atribut numeric
1e8
165057269.0
55137881.0
2998441.0
956.58
3000.0
3999591.0
1915.0
2999495.0
2999918.0
1306.0
Regular_fiber_diet
Sedentary_hours_daily
Age
Est_avg_calorie_intake
Main_meals_daily
Height
Water_daily
Weight
Physical_activity_level
Technology_time_use
Valori_minime pentru fiecare atribut numeric
1500.0
1.0
2.21
15.0
1.0
1.45
1.0
39.0
0.0
0.0
Regular_fiber_diet
Sedentary_hours_daily
Age
Est_avg_calorie_intake
Main_meals_daily
Height
Water_daily
Weight
Physical_activity_level
Technology_time_use
Valori_maxime pentru fiecare atribut numeric
3000.0
3.0
4.67
61.0
4.0
1.98
3.0
130.0
3.0
2.0
Regular_fiber_diet
Sedentary_hours_daily
Age
Est_avg_calorie_intake
Main_meals_daily
Height
Water_daily
Weight
Physical_activity_level
Technology_time_use

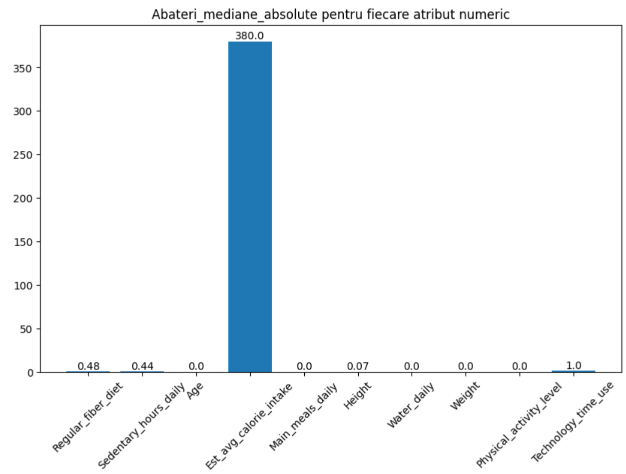


O valoare mica ar indica faptul ca datele noastre contin multi indivizi similari deci vom obtine rezultate care vor fi valide pe gama restransa de indivizi. Desi suntem tentati sa credem asta valorile nu sunt exact mici deoarece 0.53cm este o diferenta de inaltime decenta la fel si 46 in cazul varstei. La fel si in cazul altor attribute cum ar fi "Physical_activity_level" = 3 (luand in considerare ca norma e 1.4 – 2.4), 3 indica o gama larga a valorilor. O problema poate fi observata la "Regular_fiber_diet" pe care l-am observat inca de la medie.



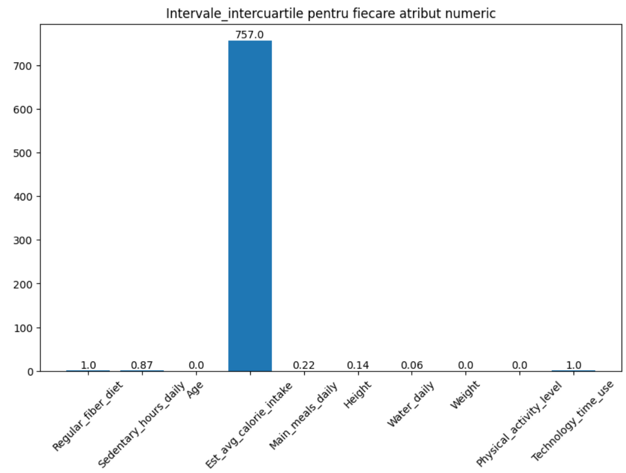


Deoarece Mediana apropiata de Medie => avem un set de date echilibrat. (set = [1, 3, 3, **6**, 7, 8, 9] Mediana = 6) Inseamna ca avem valori peste medie si sub medie in numar echilibrat.



Valorile mari ale MAD indică că datele sunt foarte dispersate în jurul medianei, adică există multe valori care sunt mult mai mari sau mult mai mici decât mediana. Pe de altă parte, valorile mici ale MAD indică că majoritatea datelor sunt aproape de mediana.

Deci observam valori foarte mici pentru multe attribute de pe grafic, in evidenta sarind cele = 0. Acest lucru se datoreaza faptului ca avem multe valori appropriate de mediana, ba chiar = cu mediana deoarece am avut multe valori eronate in acel atribut si le-am inlocuit cu media prin procedeul de Imputatie. De aici tragem concluzia ca setul nostrum de date devine usor “inaccurate” deoarece dam multor attribute, valori egale, ceea ce nu corespunde cu un studiu “accurate”. Dar lucram cu ce avem.



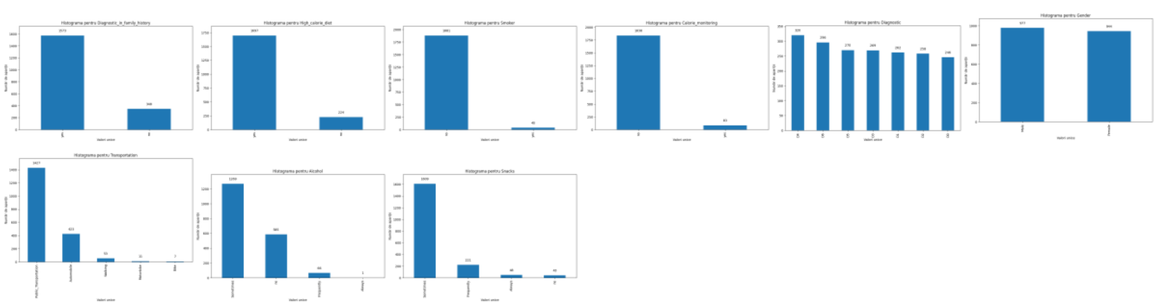
IQR = Q3 – Q1

- Q1 valoarea sub care se află 25% din date.
- Q3 valoarea sub care se află 75% din date.

Este mai puțin sensibil la valori aberante și, prin urmare, poate fi mai util.

Observam si aici multe valori = 0, lucru care a fost observant si mai sus deci atributele = 0 sunt destul de “inacurate”.

Nominale:



Covarianta:

Column1	Regular_fiber_die	Sedentary_hours_dail	Age	Est_avg_calorie_intak	Main_meals_dail	Height	Water_dail	Weight	Physical_activity_leve	Technology_time_us
Regular_fiber_diet	0.182131785	-0.014642591	0.145675	-6.357666348	0.01817218	-0.000450379	-0.002072975	-0.186180868	-0.006123996	-0.026299
Sedentary_hours_daily	-0.014642591	0.331321903	-0.28165	4.431327729	0.019912662	0.002714756	0.014345245	-0.079828476	0.01607464	0.336413512
Age	0.145675253	-0.281650036	13.04333	-20.70691097	-0.065464718	-0.010547017	0.007549445	6.681961931	-0.253123128	-0.405093279
Est_avg_calorie_intake	-6.357666348	4.431327729	-20.70691	188421.7951	-1.916366982	-2.210311005	-2.496537423	-45.49960206	0.705883464	2.52121799
Main_meals_daily	0.01817218	0.019912662	-0.065465	-1.916366982	0.369352494	0.01285703	0.006142388	0.147338985	0.048775336	0.023016301
Height	-0.000450379	0.002714756	-0.010547	-2.210311005	0.01285703	0.008687473	0.006128713	0.2073046	0.012877469	0.003408791
Water_daily	-0.002072975	0.014345245	0.007549	-2.496537423	0.006142388	0.006128713	0.175950039	0.641518631	0.052123412	0.017520255
Weight	-0.186180868	-0.079828476	6.681962	-45.49960206	0.147338985	0.2073046	0.641518631	72.68988764	0.015614697	-0.143474953
Physical_activity_level	-0.006123996	0.01607464	-0.253123	0.705883464	0.048775336	0.012877469	0.052123412	0.015614697	0.424487354	0.030824091
Technology_time_use	-0.026299	0.336413512	-0.405093	2.52121799	0.023016301	0.003408791	0.017520255	-0.143474953	0.030824091	0.455728895

Covarianța pozitivă: variabile tind sa creasca sau sa scada impreuna

Covarianta negative: una dintre variabile tinde sa creasca atunci cand cealalta scade.

Column1	Sedentary_hours_daily
Regular_fiber_diet	-0.014642591

“Regular_fiber_diet” tinde sa creasca atunci cand “Sedentary_hours_daily” scade, si invers.

3.2 - Analia VarianceTreshold

SVM:	
Initial.shape: (1921, 18)	
Threshold: 0.1	Threshold: 0.2
Reduced.shape: (1921, 15)	Reduced.shape: (1921, 11)
Hiper-Parametri: {'C': 10, 'kernel': 'rbf'}	Hiper-Parametri: {'C': 10, 'kernel': 'rbf'}

Classification Report (Threshold 0.1):

	precision	recall	f1-score	support
D0	0.81	0.73	0.77	60
D1	0.63	0.71	0.67	52
D2	0.51	0.50	0.51	42
D3	0.62	0.57	0.60	49
D4	0.81	0.62	0.70	69
D5	0.65	0.85	0.73	60
D6	0.96	0.98	0.97	53
accuracy			0.72	385
macro avg	0.71	0.71	0.71	385
weighted avg	0.73	0.72	0.72	385

Classification Report (Threshold 0.2):

	precision	recall	f1-score	support
D0	0.80	0.62	0.70	60
D1	0.55	0.60	0.57	52
D2	0.56	0.52	0.54	42
D3	0.46	0.45	0.45	49
D4	0.72	0.45	0.55	69
D5	0.51	0.80	0.62	60
D6	0.88	0.98	0.93	53
accuracy			0.63	385
macro avg	0.64	0.63	0.62	385
weighted avg	0.65	0.63	0.63	385

RandomForest:	
Initial.shape: (1921, 18)	
Threshold: 0.1	Threshold: 0.2
Reduced.shape: (1921, 15)	Reduced.shape: (1921, 11)
Hiper-Parametri: {'n_estimators': 100, 'max_depth': 20, 'max_features': 'log2'}	Hiper-Parametri: {'n_estimators': 50, 'max_depth': 10, 'max_features': 'sqrt'}

Classification Report (Threshold 0.1):

	precision	recall	f1-score	support
D0	0.94	0.75	0.83	60
D1	0.68	0.94	0.79	52
D2	0.59	0.52	0.56	42
D3	0.68	0.61	0.65	49
D4	0.77	0.67	0.71	69
D5	0.73	0.85	0.78	60
D6	0.96	0.98	0.97	53
accuracy			0.77	385
macro avg	0.76	0.76	0.76	385
weighted avg	0.77	0.77	0.76	385

Classification Report (Threshold 0.2):

	precision	recall	f1-score	support
D0	0.86	0.63	0.73	60
D1	0.67	0.81	0.73	52
D2	0.55	0.43	0.48	42
D3	0.50	0.49	0.49	49
D4	0.68	0.49	0.57	69
D5	0.59	0.87	0.70	60
D6	0.88	0.98	0.93	53
accuracy			0.68	385
macro avg	0.68	0.67	0.66	385
weighted avg	0.68	0.68	0.67	385

ExtraTrees:

Initial.shape:
(1921, 18)

Threshold: 0.1	Threshold: 0.2
Reduced.shape: (1921, 15)	Reduced.shape: (1921, 11)
Hiper-Parametri: {'n_estimators': 100, 'max_depth': 20, 'max_features': 'sqrt' }	Hiper-Parametri: {,'n_estimators': 50', max_depth': 10, 'max_features': 'None'}

Classification Report (Threshold 0.1):

```
              precision    recall  f1-score   support

          D0       0.88      0.77      0.82        60
          D1       0.68      0.87      0.76        52
          D2       0.60      0.50      0.55        42
          D3       0.69      0.71      0.70        49
          D4       0.84      0.67      0.74        69
          D5       0.69      0.83      0.76        60
          D6       0.96      0.98      0.97        53

    accuracy                           0.77       385
   macro avg       0.76      0.76      0.76       385
weighted avg       0.77      0.77      0.76       385
```

Classification Report (Threshold 0.2):

```
              precision    recall  f1-score   support

          D0       0.91      0.65      0.76        60
          D1       0.71      0.75      0.73        52
          D2       0.52      0.52      0.52        42
          D3       0.50      0.49      0.49        49
          D4       0.70      0.51      0.59        69
          D5       0.60      0.87      0.71        60
          D6       0.87      0.98      0.92        53

    accuracy                           0.68       385
   macro avg       0.69      0.68      0.67       385
weighted avg       0.70      0.68      0.68       385
```

GradientBoostedTrees:

Initial.shape:
(1921, 18)

Threshold: 0.1	Threshold: 0.2
Reduced.shape: (1921, 15)	Reduced.shape: (1921, 11)
Hiper-Parametri: {'n_estimators': 100, 'max_depth': 20, 'learning_rate': 0.1}	Hiper-Parametri: {'n_estimators': 50, 'max_depth': 10, 'learning_rate': 0.1}

Classification Report (Threshold 0.1):

```
              precision    recall  f1-score   support

          D0       0.88      0.75      0.81        60
          D1       0.72      0.88      0.79        52
          D2       0.66      0.50      0.57        42
          D3       0.62      0.71      0.67        49
          D4       0.73      0.62      0.67        69
          D5       0.74      0.85      0.79        60
          D6       0.96      0.98      0.97        53

    accuracy                           0.76       385
   macro avg       0.76      0.76      0.75       385
weighted avg       0.76      0.76      0.76       385
```

Classification Report (Threshold 0.2):

```
              precision    recall  f1-score   support

          D0       0.87      0.65      0.74        60
          D1       0.72      0.85      0.78        52
          D2       0.50      0.33      0.40        42
          D3       0.53      0.49      0.51        49
          D4       0.55      0.52      0.53        69
          D5       0.60      0.80      0.69        60
          D6       0.87      0.98      0.92        53

    accuracy                           0.67       385
   macro avg       0.66      0.66      0.65       385
weighted avg       0.67      0.67      0.66       385
```

Concluzie:
In acest caz procedura de VarianceThreshold nu a fost prea utila deoarece obtinem acc mai mica decat in cazul in care nu am aplicat procedura. Las mai jos rezultatele pentru varianta obisnuita:

Cele mai bune rezultate obtinute sunt pe varianta fara Variance Threshold, deci aici vom face si interpretarea performantelor obtinute:

SVM:

Pentru GridSearch:
Hiper-parametri: {'C': 10, 'kernel': 'rbf'}

Classification Report:

	precision	recall	f1-score	support
D0	0.85	0.77	0.81	60
D1	0.60	0.71	0.65	52
D2	0.61	0.60	0.60	42
D3	0.64	0.61	0.62	49
D4	0.84	0.62	0.72	**69**
D5	0.68	0.87	0.76	60
D6	**0.98**	**0.98**	**0.98**	53
accuracy			0.74	385
macro avg	0.74	0.74	0.73	385
weighted avg	0.75	0.74	0.74	385

acc: 0.74 > 0.72

Manual:

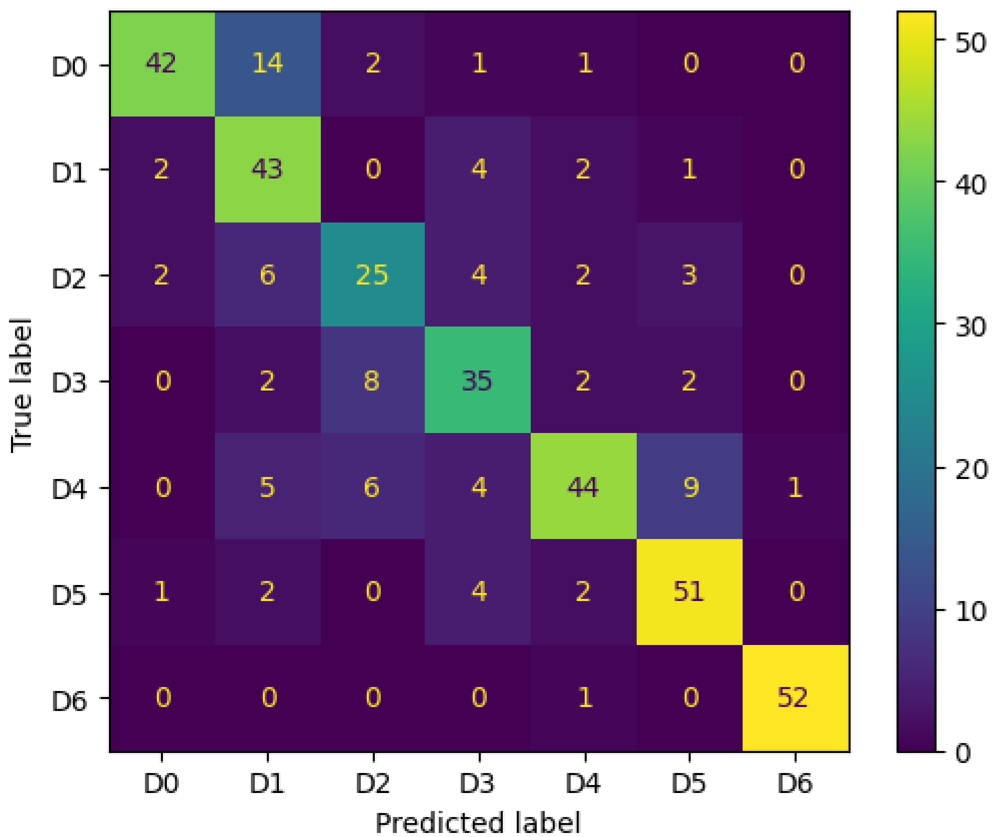
C	0.1	1	10	100
Linear acc	0.62	0.63	0.63	0.63
Poly acc	0.36	0.63	0.69	0.70
Rbf acc	0.58	0.70	0.74	0.76
Sigmoid acc	0.52	0.52	0.46	0.41

Best Hiper-Parametri: {'C': 100, 'kernel': 'rbf'}

Classification Report:

	precision	recall	f1-score	support
D0	0.89	0.70	0.79	60
D1	0.60	0.83	0.69	52
D2	0.61	0.60	0.60	42
D3	0.67	0.71	0.69	49
D4	0.81	0.64	0.72	**69**
D5	0.77	0.85	0.81	60
D6	**0.98**	**0.98**	**0.98**	53
accuracy			0.76	385
macro avg	0.76	0.76	0.75	385
weighted avg	0.77	0.76	0.76	385

```
Confusion Matrix:
```



Hiper-parametrii pot influenta negative acuratetea predictiilor ca in cazul:
`{'C': 0.1, 'kernel': 'poly'}` acc = 0.36, trebuie de considerat si complexitatea temporala care creste odata cu C, deci in cazul unui set gigantic am considera alegerea unei configuratiei:
`{'C': 1, 'kernel': 'rbf'}` unde acc 0.70 (best 0.76). Deci intelegerea comportamentului setului nostrum de date ne poate ajuta sa evitam complexitatea temporala generata de GridSearch la fel si analizarea trade-off-urilor poate imbunatati performanta.

Clasa	Sum
D0	2.38
D1	2.12
D2	1.81
D3	2.07
D4	2.17
D5	2.43
D0	2.94

Cele mai bune predictii de clasa sunt la D6, precizie = 0.98, un recall = 0.98 si scor F1 = 0.98. Apoi suntem tentati sa spunem clasa D0 deoarece are precizie = 0.89 insa avem recall mic = 0.70 si scor F1 = 0.79 (idea e sa maximizam aceste valori) iar D0 = 2.38 in timp ce D5 = 2.43. Ne asiguram de acest lucru inspectand si matricea de confuzie si observam multe exemple clasificate gresit in cazul D0.

D2 are cea mai rea preddictie.

RandomForest:

Pentru GridSearch:
Hiper-parametri: {'max_depth': None, 'max_features': 'sqrt', 'n_estimators': 100}

Classification Report:

	precision	recall	f1-score	support
D0	**0.96**	0.77	0.85	60
D1	0.67	0.94	0.78	52
D2	0.75	0.64	0.69	42
D3	0.75	0.73	0.74	49
D4	0.89	0.72	0.80	**69**
D5	0.80	0.93	0.86	60
D6	**0.96**	**0.98**	**0.97**	53
accuracy			0.82	385
macro avg	0.83	0.82	0.81	385
weighted avg	0.83	0.82	0.82	385

acc: 0.82 > 0.77

Manual:

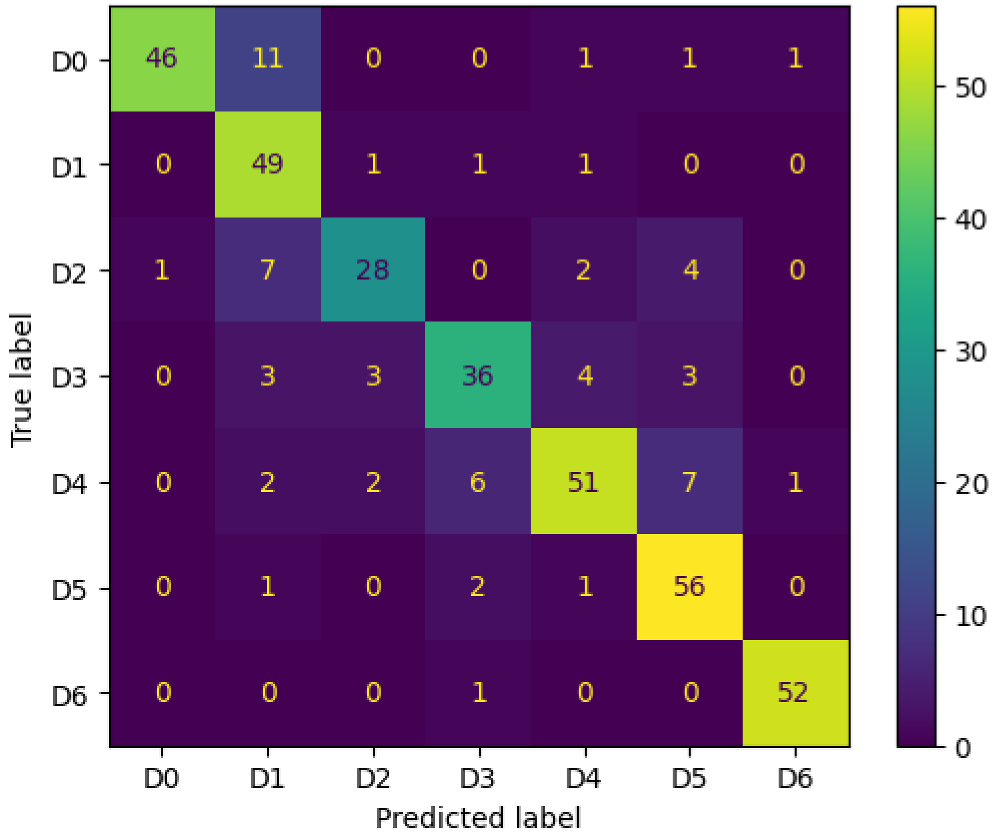
n_estimator	max_dept	max_feature	accuracy
10		sqrt	0.792207792
10		log2	0.779220779
10			0.8
10	10	sqrt	0.768831169
10	10	log2	0.763636364
10	10		0.776623377
10	20	sqrt	0.771428571
10	20	log2	0.766233766
10	20		0.794805195
10	30	sqrt	0.815584416
10	30	log2	0.78961039
10	30		0.768831169
50		sqrt	0.828571429
50		log2	0.820779221
50			0.820779221
50	10	sqrt	0.78961039
50	10	log2	0.781818182
50	10		0.81038961
50	20	sqrt	0.8
50	20	log2	0.812987013
50	20		0.828571429
50	30	sqrt	0.828571429
50	30	log2	0.823376623
50	30		0.820779221
100		sqrt	0.818181818
100		log2	0.831168831
100			0.823376623
100	10	sqrt	0.8
100	10	log2	0.8
100	10		0.781818182
100	20	sqrt	0.838961039
100	20	log2	0.81038961
100	20		0.828571429
100	30	sqrt	0.812987013
100	30	log2	0.825974026
100	30		0.815584416

Best Hiper-Param: {'n_estimators': 100, 'max_depth': 20, 'max_features': 'sqrt'}

Classification Report:

	precision	recall	f1-score	support
D0	**0.98**	0.77	0.86	60
D1	0.67	0.94	0.78	52
D2	0.82	0.67	0.74	42
D3	0.78	0.73	0.76	49
D4	0.85	0.74	0.79	**69**
D5	0.79	0.93	0.85	60
D6	0.96	**0.98**	**0.97**	53
accuracy			0.83	385
macro avg	0.84	0.82	0.82	385
weighted avg	0.84	0.83	0.83	385

```
Confusion Matrix:
```



Aici observam o distributie mai echilibrata a acuratetii decat in cazul SVM. Acc_min aprox 0.76 iar acc_max = 0.83. Analizand acest lucru putem allege o configuratie a hiperparametrilor cu un n_estimators si max_depth mai mici precum:
`{'max_depth': None, 'max_features': 'None', 'n_estimators': 10} unde acc = 0.8`

Obtinem un trade-off convenabil intre complexitate si performanta. Deci si in acest caz hiper-parametrii pot afecta considerabil performanta in dependenta de ce urmarim (pe un set mare, viteza iar pe un set mic acuratete), in cazul setului nostru unde avem putine date putem sa urmarim acuratetea rezultatului si sa evitam costul unui GridSearch.

Clasa	Sum
D0	2.61
D1	2.39
D2	2.23
D3	2.27
D4	2.38
D5	2.57
D6	2.91

Cele mai bune predictii de clasa sunt D6 si D0 iar cele mai slabe sunt D2 si D3. Observam astfel o imbunatatire considerabila fata de SVM.

ExtraTrees:

```
Pentru GridSearch:
Hiper-parametri: {'max_depth': 30, 'max_features': 'log2', 'n_estimators': 100}
```

```
Classification Report:
              precision    recall  f1-score   support

          D0       0.96      0.77      0.85        60
          D1       0.68      0.85      0.75        52
          D2       0.68      0.67      0.67        42
          D3       0.76      0.76      0.76        49
          D4       0.89      0.78      0.83        69
          D5       0.75      0.85      0.80        60
          D6       0.98      0.98      0.98        53

    accuracy                           0.81       385
   macro avg       0.81      0.81      0.81       385
weighted avg       0.82      0.81      0.81       385
```

```
acc: 0.81 > 0.77
```

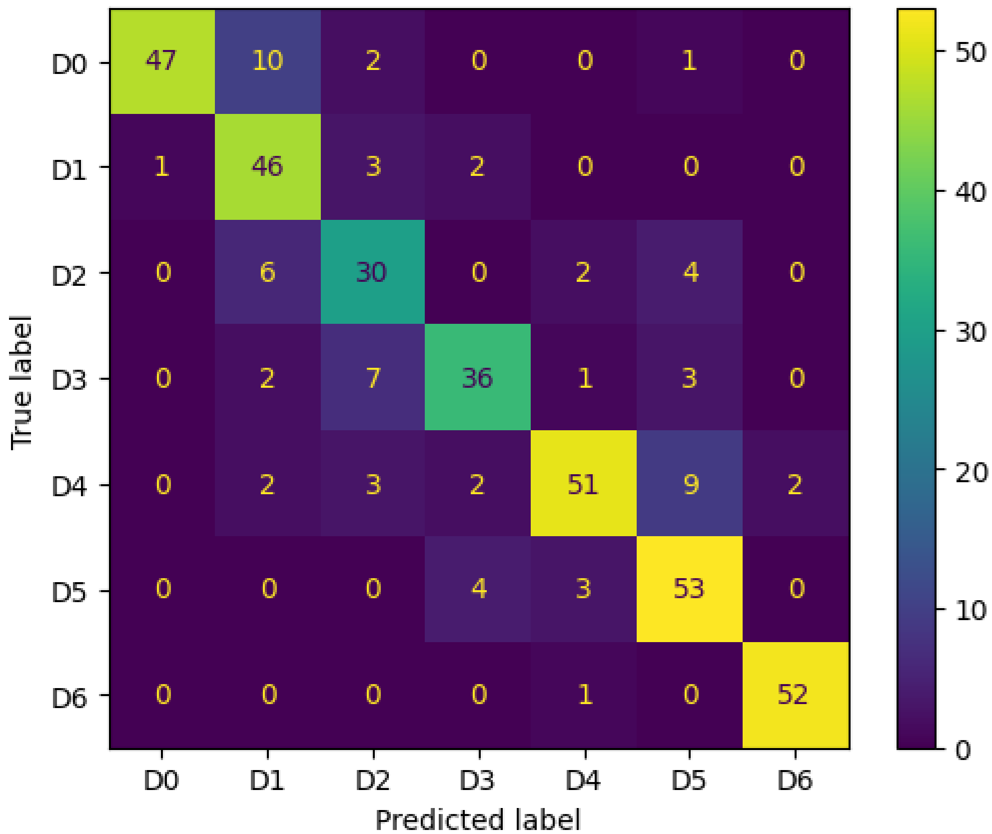
```
Manual:
```

n_estimator	max_dept	max_feature	accurac
10		sqrt	0.7922078
10		log2	0.774026
10			0.7948052
10	10	sqrt	0.7298701
10	10	log2	0.7064935
10	10		0.7506494
10	20	sqrt	0.7714286
10	20	log2	0.7948052
10	20		0.8051948
10	30	sqrt	0.7896104
10	30	log2	0.8051948
10	30		0.7818182
50		sqrt	0.8155844
50		log2	0.812987
50			0.8155844
50	10	sqrt	0.7506494
50	10	log2	0.7558442
50	10		0.7558442
50	20	sqrt	0.8207792
50	20	log2	0.8103896
50	20		0.8103896
50	30	sqrt	0.8103896
50	30	log2	0.8025974
50	30		0.8103896
100		sqrt	0.8077922
100		log2	0.8077922
100			0.812987
100	10	sqrt	0.7662338
100	10	log2	0.7636364
100	10		0.7766234
100	20	sqrt	0.8103896
100	20	log2	0.8181818
100	20		0.8103896
100	30	sqrt	0.812987
100	30	log2	0.8103896
100	30		0.8207792

```
Best Hiper-Param: {'n_estimators': 50, 'max_depth': 20, 'max_features': 'sqrt'}
```

```
Classification Report:
              precision    recall  f1-score   support

          D0       0.98      0.78      0.87        60
          D1       0.70      0.88      0.78        52
          D2       0.67      0.71      0.69        42
          D3       0.82      0.73      0.77        49
          D4       0.88      0.74      0.80        69
          D5       0.76      0.88      0.82        60
          D6       0.96      0.98      0.97        53

    accuracy                           0.82       385
   macro avg       0.82      0.82      0.81       385
weighted avg       0.83      0.82      0.82       385
```

```
Confusion Matrix:
```



Se poate observa faptul ca n_estimatori = 50 obtine cel mai convenabil trade-off deoarece pentru:

```
{'n_estimators': 50, 'max_depth': 20, 'max_features': 'sqrt'} acc = 0.8207
{'n_estimators': 50, 'max_depth': 20, 'max_features': 'log2'} acc = 0.8103
{'n_estimators': 50, 'max_depth': 20, 'max_features': 'None'} acc = 0.8103
```

```
{'n_estimators': 100, 'max_depth': 30, 'max_features': 'sqrt'} acc = 0.8129
{'n_estimators': 100, 'max_depth': 30, 'max_features': 'log2'} acc = 0.8103
{'n_estimators': 100, 'max_depth': 30, 'max_features': 'None'} acc = 0.8207
```

Observam aproape aceleasi valori => mai rentabil sa folosim:

```
{'n_estimators': 50, 'max_depth': 20} deoarece optinem performanta crescuta la un
cost mai mic.
```

Clasa	Sum
D0	2.63
D1	2.36
D2	2.07
D3	2.32
D4	2.34
D5	2.42
D6	2.91

Cele mai bune predictii de clasa sunt D6 si D0 iar cele mai slabe sunt D2 si D3. Observam comportament similar cu RandomForest deoarece avem diferenta acc_max – acc_min neglijabila.

GradientBoostedTrees:

Pentru GridSearch:
Hiper-parametri: {'n_estimators': 50, 'max_depth': 10, 'learning_rate': 0.1}

Classification Report:

	precision	recall	f1-score	support
D0	0.94	0.75	0.83	60
D1	0.74	0.88	0.81	52
D2	0.67	0.57	0.62	42
D3	0.77	0.73	0.75	49
D4	0.82	0.74	0.78	**69**
D5	0.76	0.95	0.84	60
D6	**0.95**	**0.98**	**0.96**	53
accuracy			0.81	385
macro avg	0.81	0.80	0.80	385
weighted avg	0.81	0.81	0.81	385

acc: 0.81 > 0.76
Manual:

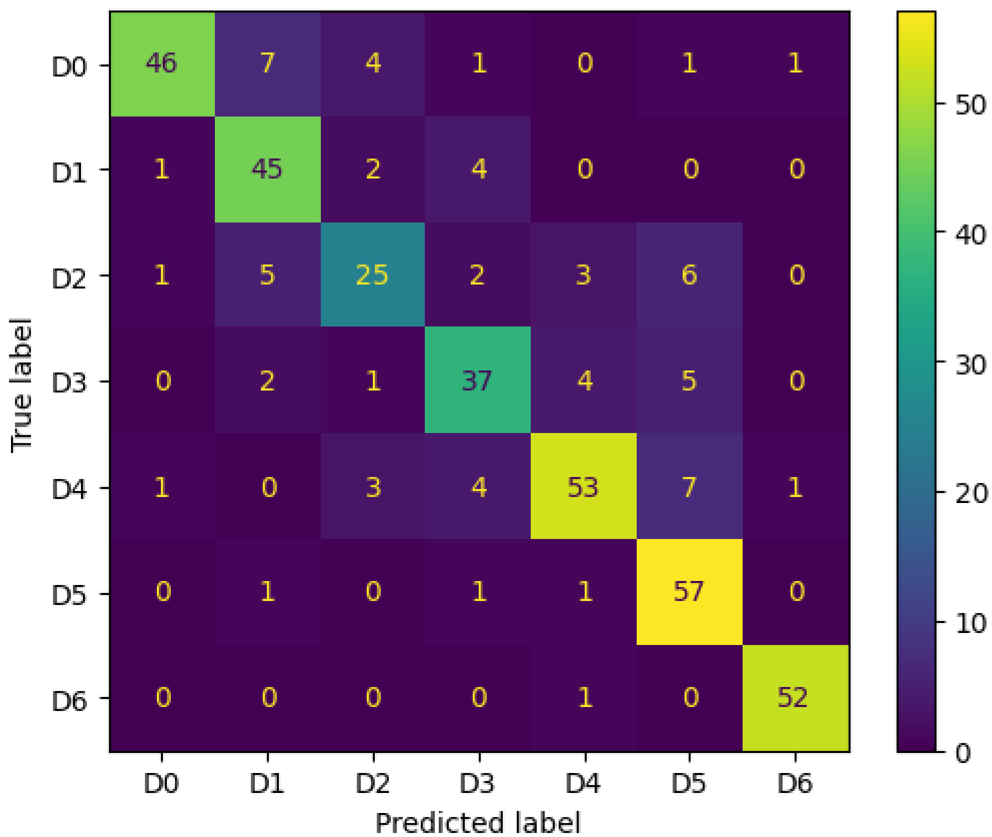
n_estimator	max_dept	learning_rat	accurac
10	3	0.1	0.6831169
10	3	0.01	0.6415584
10	3	0.001	0.6467532
10	10	0.1	0.774026
10	10	0.01	0.7584416
10	10	0.001	0.7506494
10	20	0.1	0.7948052
10	20	0.01	0.7714286
10	20	0.001	0.7636364
10	30	0.1	0.7948052
10	30	0.01	0.7714286
10	30	0.001	0.7636364
50	3	0.1	0.7272727
50	3	0.01	0.6623377
50	3	0.001	0.6337662
50	10	0.1	0.8077922
50	10	0.01	0.7662338
50	10	0.001	0.7506494
50	20	0.1	0.8077922
50	20	0.01	0.7844156
50	20	0.001	0.7714286
50	30	0.1	0.812987
50	30	0.01	0.7844156
50	30	0.001	0.7714286
100	3	0.1	0.7948052
100	3	0.01	0.6779221
100	3	0.001	0.6415584
100	10	0.1	0.8181818
100	10	0.01	0.7766234
100	10	0.001	0.7558442
100	20	0.1	0.8077922
100	20	0.01	0.7974026
100	20	0.001	0.7688312
100	30	0.1	0.8103896
100	30	0.01	0.7974026
100	30	0.001	0.7688312

Best Hiper-Param: {'n_estimators': 100, 'max_depth': 10, 'learning_rate': 0.1}

Classification Report:

	precision	recall	f1-score	support
D0	0.94	0.77	0.84	60
D1	0.75	0.87	0.80	52
D2	0.71	0.60	0.65	42
D3	0.76	0.76	0.76	49
D4	0.85	0.77	0.81	**69**
D5	0.75	0.95	0.84	60
D6	**0.96**	**0.98**	**0.97**	53
accuracy			0.82	385
macro avg	0.82	0.81	0.81	385
weighted avg	0.82	0.82	0.82	385

Confusion Matrix:



Observam ca acuratetea predictiei creste odata cu cresterea learning_rate-ului, lucru surprinzator deoarece, un learning rate mai mare va face ca algoritmul sa convearga mai rapid deci obtinem performante mai bune. In schimb, nu observam liniaritate intre cresterea estimatorilor si a max_dept-ului deoarece avem comportamente de genul:

```
{'n_estimators': 100, 'max_depth': 10, 'learning_rate': 0.1} acc = 0.818
{'n_estimators': 100, 'max_depth': 20, 'learning_rate': 0.1} acc = 0.807
{'n_estimators': 100, 'max_depth': 30, 'learning_rate': 0.1} acc = 0.810
```

Trade-off gasit e la mijloc de ex:

```
{'n_estimators': 50, 'max_depth': 20, 'learning_rate': 0.1} acc = 0.807
{'n_estimators': 50, 'max_depth': 30, 'learning_rate': 0.1} acc = 0.812
```

Aici putem concluziona faptul ca ar fi mai sigur folosirea GridSearch.

Clasa	Sum
D0	2.55
D1	2.42
D2	1.96
D3	2.28
D4	2.43
D5	2.54
D6	2.91

Cele mai bune predictii de clasa sunt D6 si D0 iar cele mai slabe sunt D2 si D3. Aici diferenta acc_max – acc_min este mai mare decat la RandomForest si ExtraTrees deci, cum am mai spus putem prefera strategia GridSearch.

Dupa observatiile de la IQR:
Decid sa elimin Age, Weight, Physical_activity_level

```
dataset_numeric = dataset_numeric.drop("Age", axis=1)
dataset_numeric = dataset_numeric.drop("Weight", axis=1)
dataset_numeric = dataset_numeric.drop("Physical_activity_level", axis=1)
```

SVM:

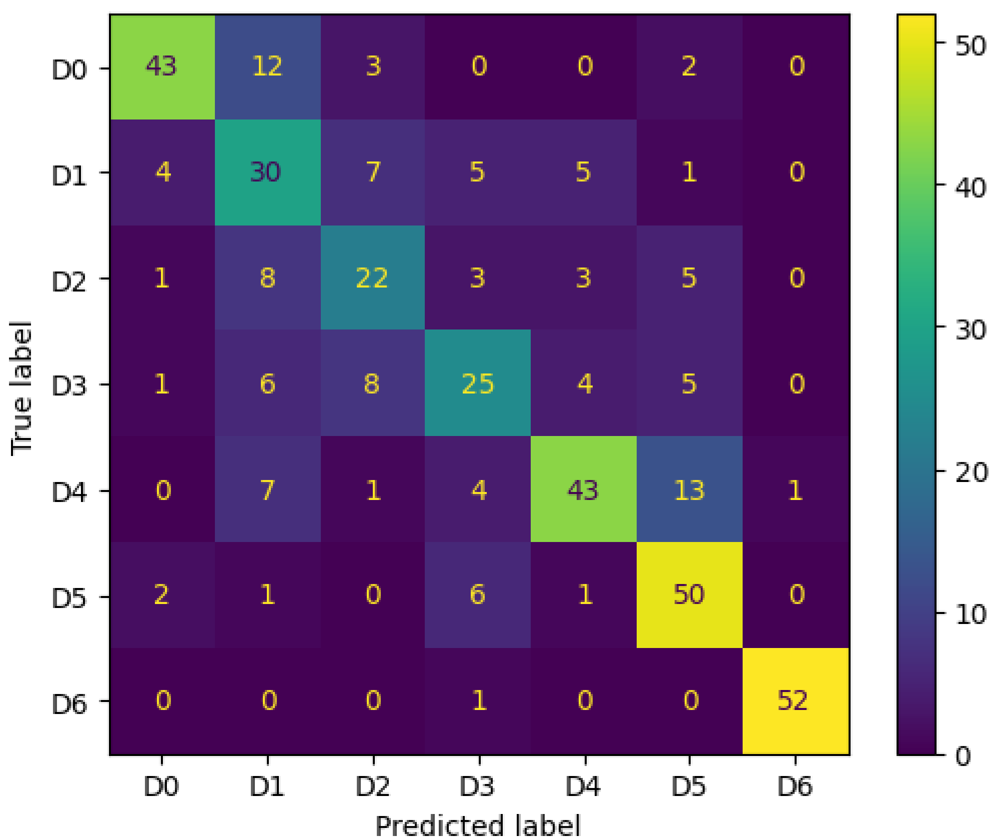
```
{'C': 10, 'kernel': 'rbf'}
```

Classification Report:

	precision	recall	f1-score	support
D0	0.84	0.72	0.77	60
D1	0.47	0.58	0.52	52
D2	0.54	0.52	0.53	42
D3	0.57	0.51	0.54	49
D4	0.77	0.62	0.69	69
D5	0.66	0.83	0.74	60
D6	0.98	0.98	0.98	53
accuracy			0.69	385
macro avg	0.69	0.68	0.68	385
weighted avg	0.70	0.69	0.69	385

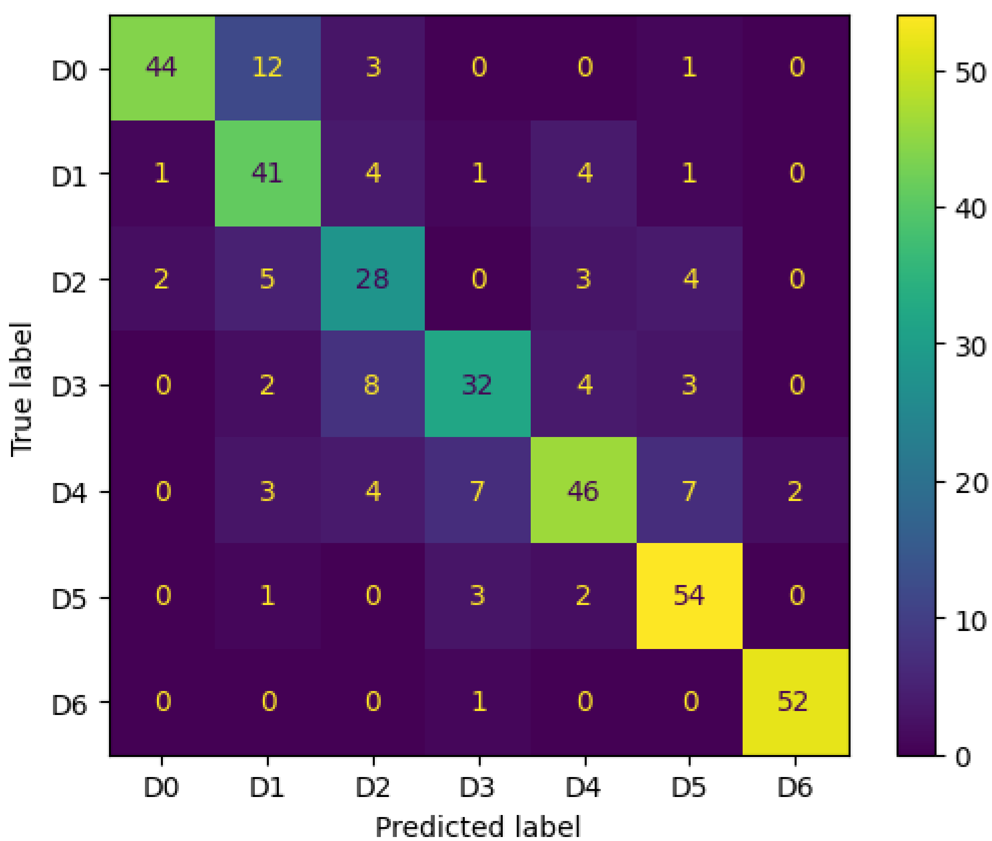
acc: 0.69 < 0.72 < 0.74

Confusion Matrix:



RandomForest:

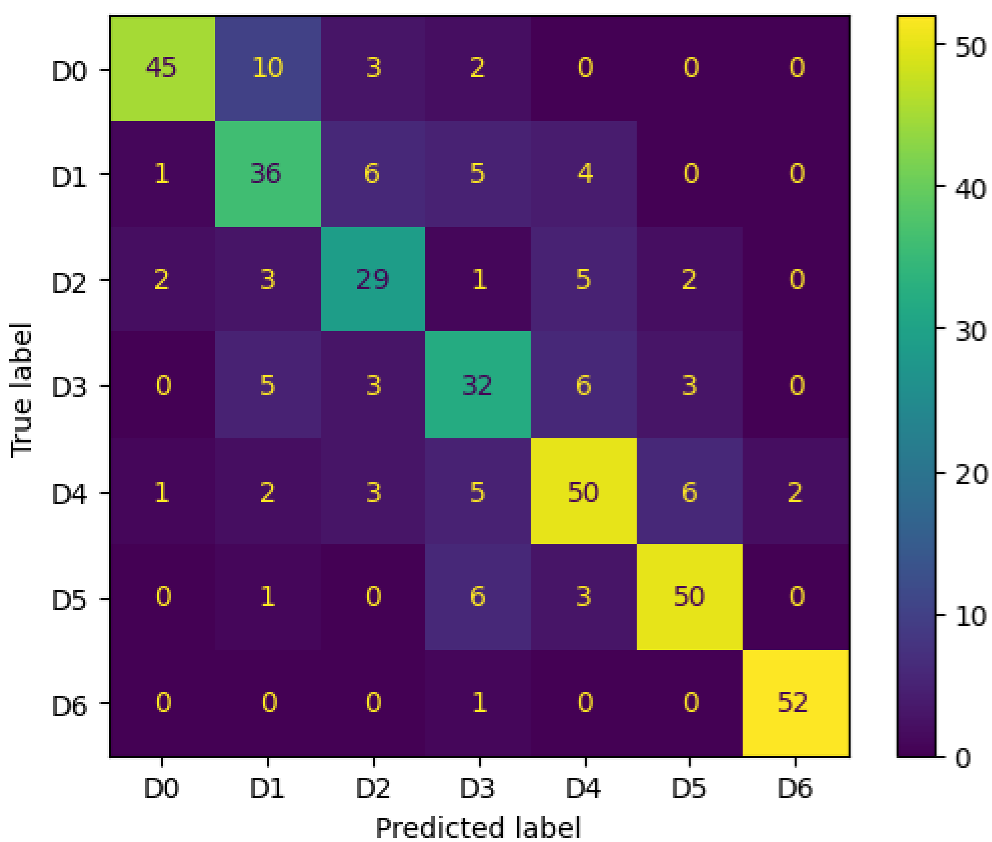
```
{'max_depth': None, 'max_features': 'sqrt', 'n_estimators': 100}


Classification Report:
              precision    recall  f1-score   support

          D0       0.94      0.73      0.82        60
          D1       0.64      0.79      0.71        52
          D2       0.60      0.67      0.63        42
          D3       0.73      0.65      0.69        49
          D4       0.78      0.67      0.72        69
          D5       0.77      0.90      0.83        60
          D6       0.96      0.98      0.97        53

    accuracy                           0.77       385
   macro avg       0.77      0.77      0.77       385
weighted avg       0.78      0.77      0.77       385

acc: 0.77 = 0.77 < 0.82

Confusion Matrix:
```



ExtraTrees:

```
{'max_depth': 30, 'max_features': None, 'n_estimators': 50}
```

```
Classification Report:
              precision    recall  f1-score   support

          D0       0.92      0.75      0.83        60
          D1       0.63      0.69      0.66        52
          D2       0.66      0.69      0.67        42
          D3       0.62      0.65      0.63        49
          D4       0.74      0.72      0.73        69
          D5       0.82      0.83      0.83        60
          D6       0.96      0.98      0.97        53

    accuracy                           0.76       385
   macro avg       0.76      0.76      0.76       385
weighted avg       0.77      0.76      0.77       385

acc: 0.76 < 0.77 < 0.81

Confusion Matrix:
```



GradientBoostedTrees:

```
{'learning_rate': 0.1, 'max_depth': 20, 'n_estimators': 100}
```

Classification Report:

	precision	recall	f1-score	support
D0	0.84	0.70	0.76	60
D1	0.57	0.65	0.61	52
D2	0.65	0.62	0.63	42
D3	0.64	0.59	0.62	49
D4	0.78	0.67	0.72	69
D5	0.75	0.95	0.84	60
D6	0.95	0.98	0.96	53
accuracy			0.74	385
macro avg	0.74	0.74	0.73	385
weighted avg	0.75	0.74	0.74	385

acc: 0.74 < 0.76 < 0.81

Confusion Matrix: