Tema 1 Învăţare Automată
Clasificare Imagini FashionMNIST

Alexandru Licuriceanu
alicuriceanu@stud.acs.upb.ro

1 Cerinţa 4.1 - Extragerea de atribute

Pentru extragea de atribute, am construit un flux bazat pe două metode: Histogram of Oriented Gradients (HoG) şi Principal Component Analysis (PCA). Am folosit acelaşi flux pentru ambele seturi de date, dar cu valori diferite pentru numărul de componente la PCA.

- HoG - L-am ales pentru capabilitatea sa de a captura informaţii legate de contur, în speranţa că vor fi eliminate informaţiile redundante, iar algoritmul de clasificare se va concentra pe forma obiectelor de clasificat.
- PCA - L-am ales pentru a reduce dimensionalitatea datelor (şi implicit resursele computaţionale folosite şi timpul de rulare), păstrând componentele esențiale care contribuie cel mai mult la variația din setul de date. Pentru această etapa, am găsit şi am ales numărul de componente care păstrează varianţa setului de date la 95%.

După aplicarea HoG, setul de date are 1296 de atribute, peste care am aplicat PCA păstrând 95% varianţa, de unde au rezultat, în final, 354 de atribute, aşadar o reducere semnificativă a numarului de atribute după care se va face clasificarea.

2 Cerinţa 4.2 - Vizualizarea atributelor extrase

2.1 Cerinţa 4.2.1 - Analiza echilibrului de clase

Am utilizat grafice de bare pentru a vizualiza distribuţia pe clase, de unde se observă echilibrul claselor, atât pe setul de date de antrenare, cât şi pe cel de testare:

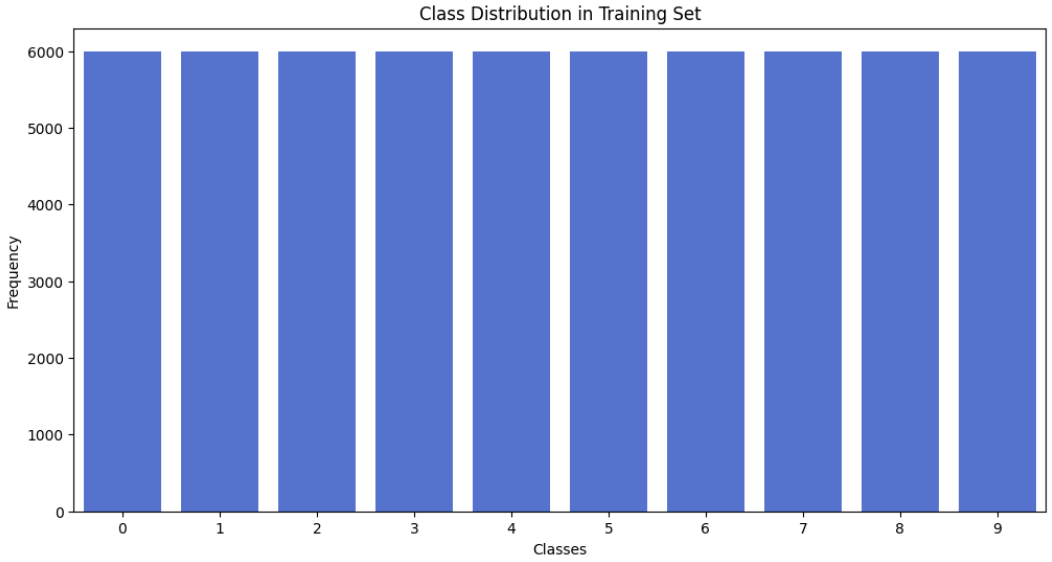

Figure 1: Distribuţia claselor în setul de antrenare.

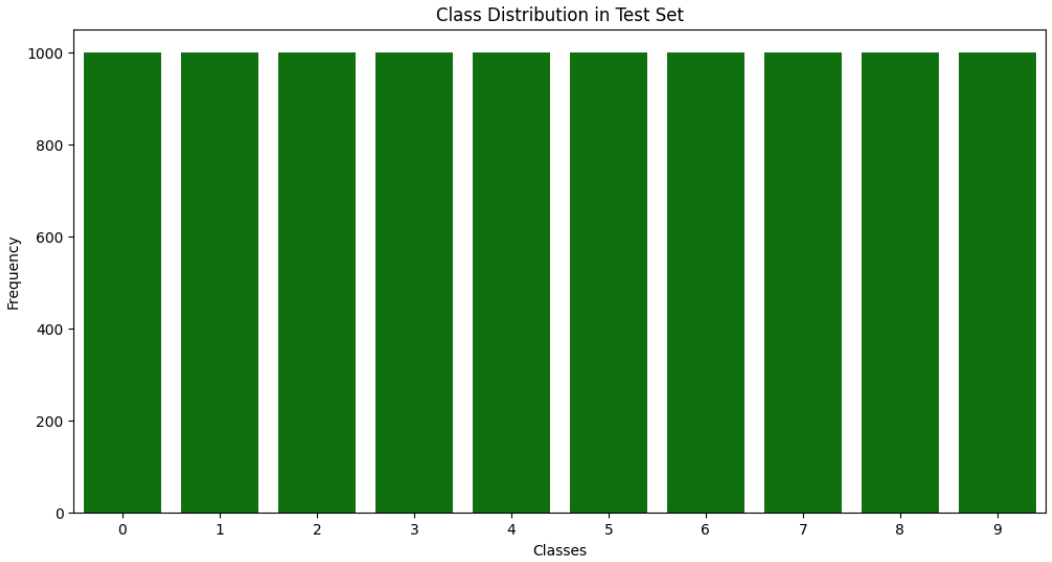

Figure 2: Distribuţia claselor în setul de testare.

2.2 Cerinţa 4.2.2 - Vizualizarea efectului de extragere a atributelor

Am afişat câte o imagine din fiecare clasă după aplicarea HoG, care ajută la capturarea trăsăturilor distinctive ale fiecărei clase, în special marignile obiectelor, care sunt destul de diferite în cadrul articolelor vestimentare şi pot ajuta clasificatorul să înveţe trăsăturile fiecărei clase.

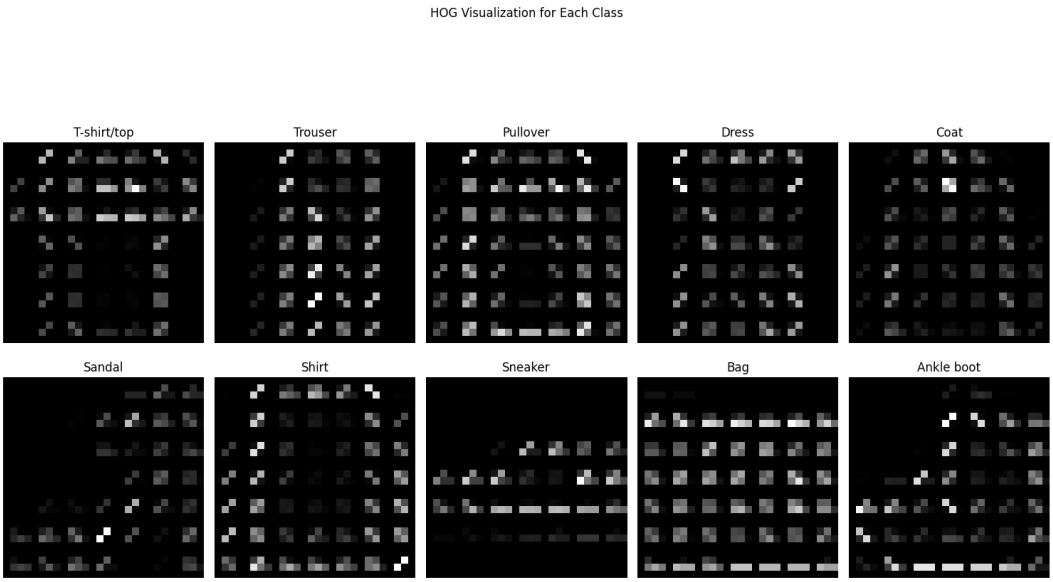

Figure 3: HoG aplicat pe câte o imagine din fiecare clasă.

Pentru PCA, mai întâi am calculat numărul de componente necesare pentru a păstra 95% varianţă a setului de date (am ales valoarea de 95% drept un compromis între acurateţea de clasificare şi cantitatea de date de procesat de modelele de învăţare automată, implicit şi timpul de rulare), apoi am afişat câte o imagine din fiecare clasă, înainte şi după reconstrucţia imaginilor.

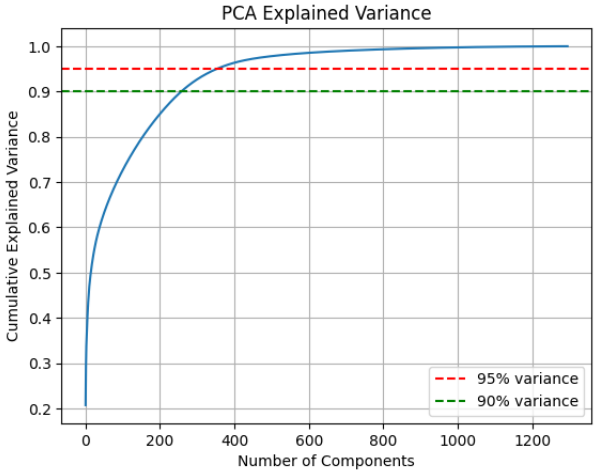

Figure 4: Număr de componente vs. varianţă.

```
Number of components for 95% variance: 354
Number of components for 90% variance: 259
```

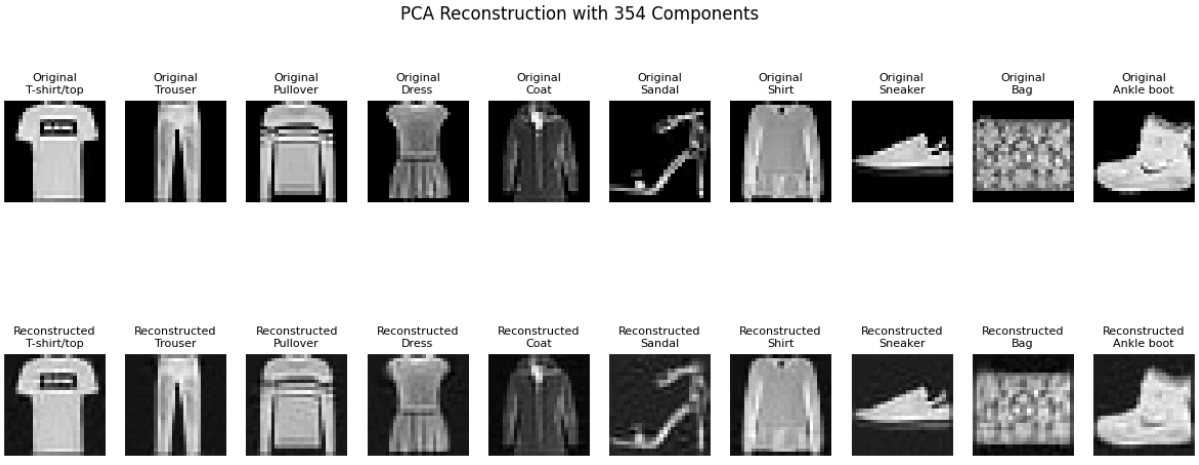

Figure 5: Imagini reconstruite cu 354 de componente.

Utilitatea în a folosi PCA este, desigur, reducerea numărului de atribute, ceea ce a scăzut complexitatea computațională a antrenării modelelor de învățare automată, făcând procesul mai rapid și mai eficient.

3 Cerinţa 4.3 - Standardizarea şi selecţia atributelor

Pentru preprocesarea datelor, am utilizat StandardScaler pentru a uniformiza valorile numerice ale atributelor. Această etapă este importantă deoarece datele pot conține atribute cu scări de valori diferite, ceea ce poate afecta performanța clasificatorilor precum SVM.

Pentru reducerea dimensionalității setului de date am aplicat SelectPercentile, care selectează cele mai relevante atribute pe baza scorului ANOVA. Am ales să selectez primele 50% cele mai importante atribute, care au cea mai mare influență asupra predicţiei, eliminând pe cele redundante sau irelevante, de unde am ajuns la 177 de atribute cu care se face clasificarea. De asemenea, am afişat şi primele 20 atribute cu cea mai mare relevanţă pentru clasificare, în funcţie de scorul ANOVA:

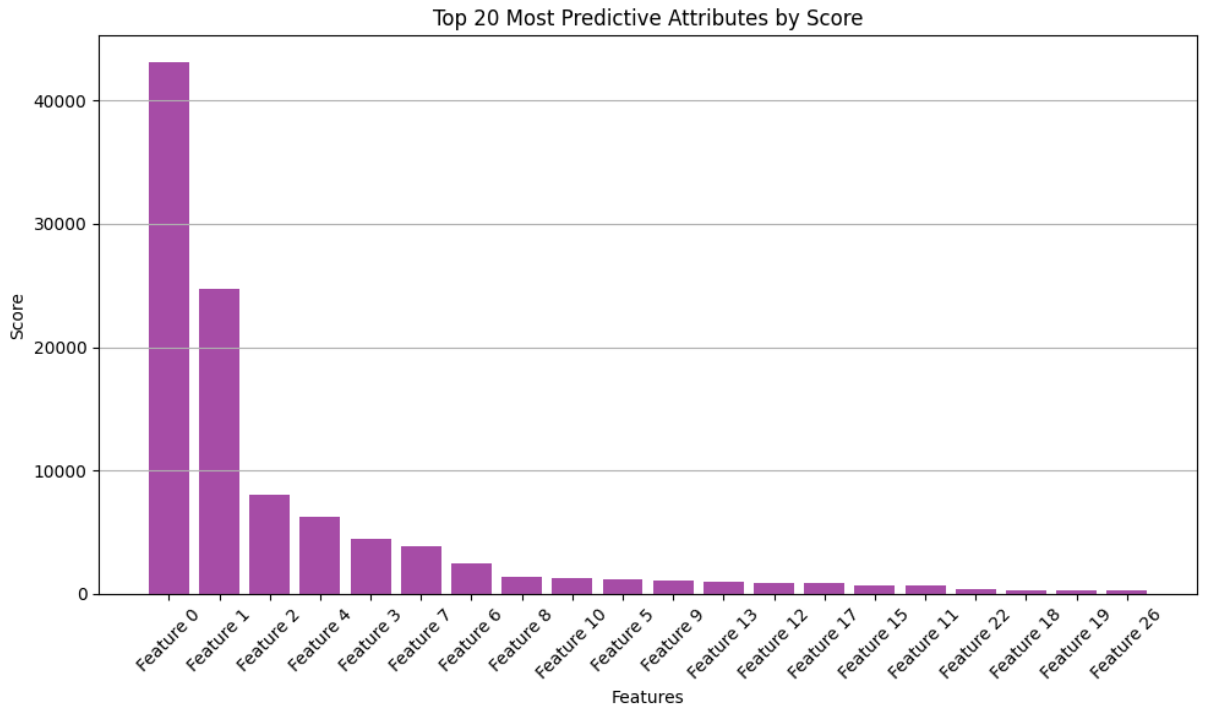

Figure 6: Primele 20 cele mai predictive atribute.

4 Cerinţa 4.4 - Modele de învăţare automată

Pentru antrenarea celor 4 modele de învăţare automată, am folosit setul de date cu primele 50% atributele selectate pe baza scorului ANOVA, de la pasul anterior. Pentru căutarea hiperparametrilor am folosit RandomizedSearchCV.

4.1 Regresie Logistică

```
C = 10
multi_class = "multinomial"
solver = "lbfgs"
```

O valoare mare a lui C înseamnă o regularizare mai slabă, ceea ce permite modelului să se potrivească mai bine cu datele de antrenare, dar poate duce la overfitting. Hiperparametrul "multi_class" este setat să folosească funcţia softmax, care calculează probabilitățile pentru toate clasele simultan și apoi alege clasa cu probabilitatea cea mai mare, "solver" este algoritmul utilizat pentru a minimiza funcția de cost, iar alegerea solver-ului influențează viteza de convergență și precizia modelului.

Table 1: Clasificare regresie logistică.

Class	Precision	Recall	F1-Score	Support
T-shirt/top	0.82	0.84	0.83	1000
Trouser	0.97	0.96	**0.97**	1000
Pullover	0.81	0.78	0.80	1000
Dress	0.87	0.86	0.87	1000
Coat	0.78	0.81	0.79	1000
Sandal	0.95	0.95	0.95	1000
Shirt	0.65	0.63	0.64	1000
Sneaker	0.92	0.95	0.93	1000
Bag	0.96	0.97	**0.97**	1000
Ankle boot	0.97	0.94	0.96	1000
Accuracy			0.87	10000
Macro avg	0.87	0.87	0.87	10000
Weighted avg	0.87	0.87	0.87	10000

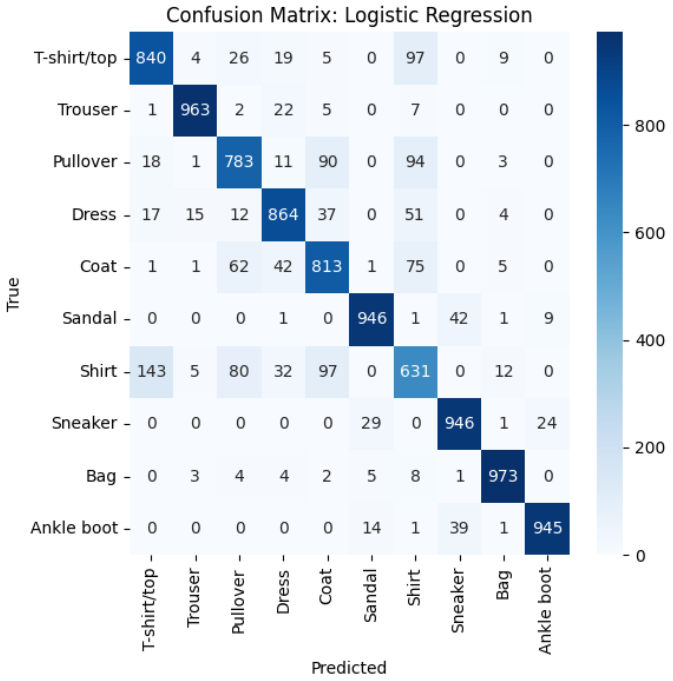

Figure 7: Matricea de confuzie pentru regresia logistică.

4.2 SVM

```
C = 1
kernel = "rbf"
```

La fel ca la regresia logistică, factorul de regularizare C înseamnă o penalizare mai mică pentru erorile de clasificare, ceea ce permite modelului să învețe mai bine datele de antrenare, dar poate duce la overfitting, iar hiperparametrul "kernel" este setat pe "rbf" (Radial Basis Function), care permite modelului să găsească frontiere non-liniare, fiind util când datele nu sunt liniar separabile.

Table 2: Clasificare SVM

Class	Precision	Recall	F1-Score	Support
T-shirt/top	0.85	0.86	0.86	1000
Trouser	0.99	0.96	**0.98**	1000
Pullover	0.85	0.83	0.84	1000
Dress	0.89	0.90	0.89	1000
Coat	0.84	0.85	0.85	1000
Sandal	0.97	0.96	0.97	1000
Shirt	0.71	0.71	0.71	1000
Sneaker	0.93	0.97	0.95	1000
Bag	0.97	0.98	**0.98**	1000
Ankle Boot	0.98	0.96	0.97	1000
Accuracy			**0.90**	10000
Macro avg	0.90	0.90	0.90	10000
Weighted avg	0.90	0.90	0.90	10000

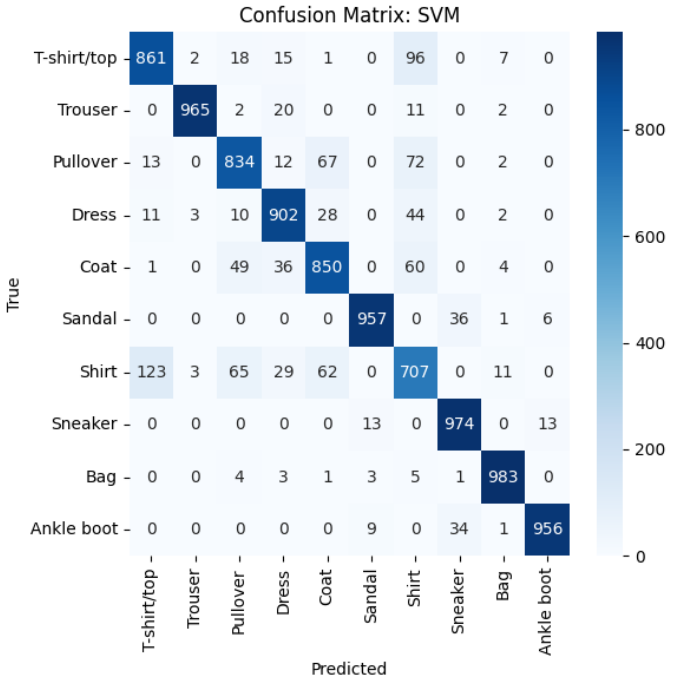

Figure 8: Matricea de confuzie pentru SVM.

4.3 Random Forest

```
n_estimators = 200
max_features = "sqrt"
max_depth = 70
```

Hiperparametrul "n_estimators" definește numărul de arbori în pădurea aleatoare, o valoare mai mare duce la un model mai robust, dar cu costuri computaționale mai mari. max_features = "sqrt" indică faptul că pentru fiecare arbore, se va alege un subset aleatoriu de atribute (rădăcina pătrată din numărul total de atribute), ceea ce poate preveni overfitting-ul, reducând corelaţia dintre arbori. Parametrul "max_depth" controlează adâncimea maximă a fiecărui arbore, însă o valoare prea mare poate duce la overfitting.

Table 3: Clasificare Random Forest.

Class	Precision	Recall	F1-Score	Support
T-shirt/top	0.81	0.85	0.83	1000
Trouser	0.99	0.95	**0.97**	1000
Pullover	0.81	0.79	0.80	1000
Dress	0.85	0.89	0.87	1000
Coat	0.78	0.82	0.80	1000
Sandal	0.94	0.94	0.94	1000
Shirt	0.68	0.60	0.64	1000
Sneaker	0.91	0.93	0.92	1000
Bag	0.94	0.97	0.95	1000
Ankle boot	0.95	0.94	0.95	1000
Accuracy			0.87	10000
Macro avg	0.87	0.87	0.87	10000
Weighted avg	0.87	0.87	0.87	10000

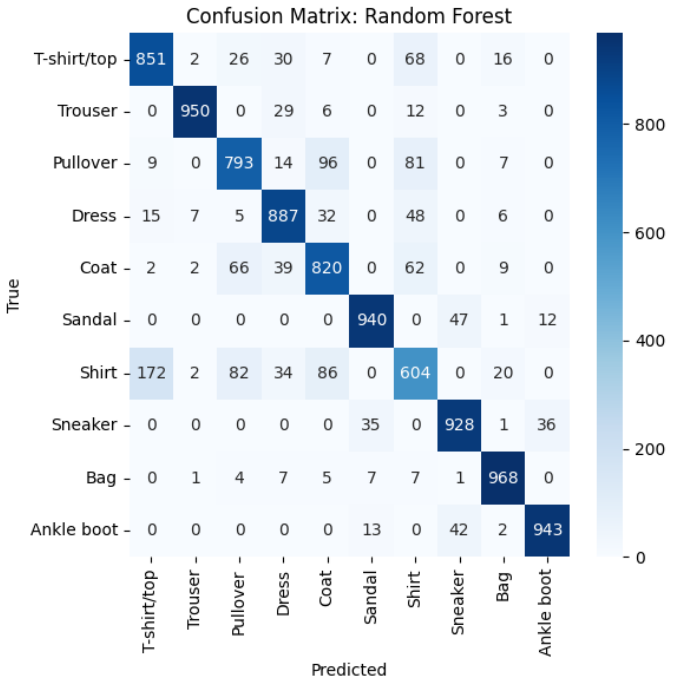

Figure 9: Matricea de confuzie pentru Random Forest.

4.4 Gradient Boosting

```
n_estimators = 200
max_depth = 10
learning_rate = 0.01
```

Asemănător ca la Random Forest, "n_estimators" şi "max_depth" controlează numărul de arbori şi respectiv adâncimea maximă a unui arbore, iar "learning_rate" controlează cât de mult contribuie fiecare arbore la ajustarea predicțiilor modelului şi impactează timpul de convergenţă al modelului.

Table 4: Clasificare Gradient Boosting.

Class	Precision	Recall	F1-Score	Support
T-shirt/top	0.83	0.83	0.83	1000
Trouser	0.98	0.95	**0.96**	1000
Pullover	0.79	0.79	0.79	1000
Dress	0.86	0.86	0.86	1000
Coat	0.75	0.79	0.77	1000
Sandal	0.94	0.93	0.93	1000
Shirt	0.65	0.63	0.64	1000
Sneaker	0.90	0.94	0.92	1000
Bag	0.95	0.95	0.95	1000
Ankle boot	0.96	0.94	0.95	1000
Accuracy			0.86	10000
Macro avg	0.86	0.86	0.86	10000
Weighted avg	0.86	0.86	0.86	10000

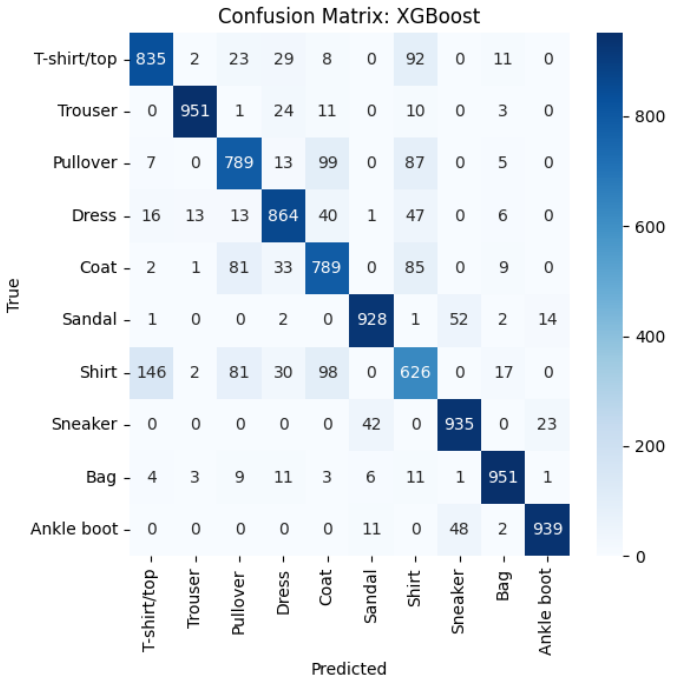

Figure 10: Matricea de confuzie pentru Gradient Boosting.

Am observat că fiecare algoritm clasifică cel mai bine imagini din clasa "Trouser" (sau "Bag" în cazul SVM), probabil pentru că imaginile cu pantaloni (sau genţi) au forme complet diferite de alte clase şi nu există vreo asemănare cu alte articole vestimentare cum ar putea exista între "Pullover" şi "Shirt", de exemplu. Per total, cea mai bună clasificare este facută de SVM, cu o acurateţe de 90%.

Tema 1 Învăţare Automată
Clasificare Imagini Fruits-360

Alexandru Licuriceanu
alicuriceanu@stud.acs.upb.ro

1 Cerinţa 4.1 - Extragerea de atribute

Pentru extragea de atribute, am construit un flux bazat pe două metode: Histogram of Oriented Gradients (HoG) şi Principal Component Analysis (PCA). Am folosit acelaşi flux pentru ambele seturi de date, dar cu valori diferite pentru numărul de componente la PCA.

- HoG - L-am ales pentru capabilitatea sa de a captura informaţii legate de contur, în speranţa că vor fi eliminate informaţiile redundante, iar algoritmul de clasificare se va concentra pe forma obiectelor de clasificat.
- PCA - L-am ales pentru a reduce dimensionalitatea datelor (şi implicit resursele computaţionale folosite şi timpul de rulare), păstrând componentele esențiale care contribuie cel mai mult la variația din setul de date. Pentru această etapa, am găsit şi am ales numărul de componente care păstrează varianţa setului de date la 95%.

După aplicarea HoG, setul de date are 4356 de atribute, peste care am aplicat PCA păstrând 95% varianţa, de unde au rezultat, în final, 1067 de atribute, aşadar o reducere semnificativă a numarului de atribute după care se va face clasificarea.

2 Cerinţa 4.2 - Vizualizarea atributelor extrase

2.1 Cerinţa 4.2.1 - Analiza echilibrului de clase

Am utilizat grafice de bare pentru a vizualiza distribuţia pe clase, de unde se observă că seturile de date nu au clasele echilibrate, însă distribuţia valorilor este la fel pentru ambele seturi de date.

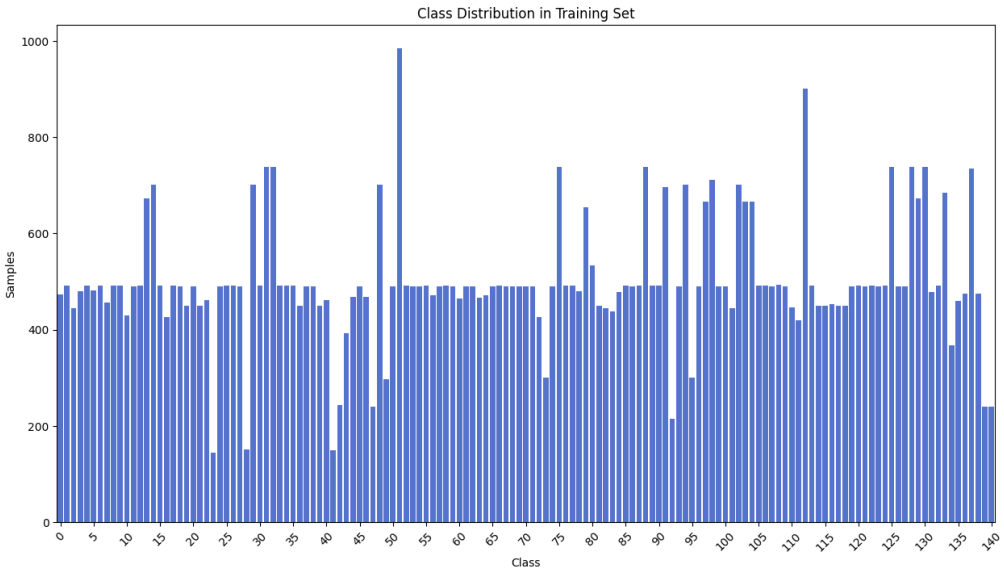

Figure 1: Distribuţia claselor în setul de antrenare.

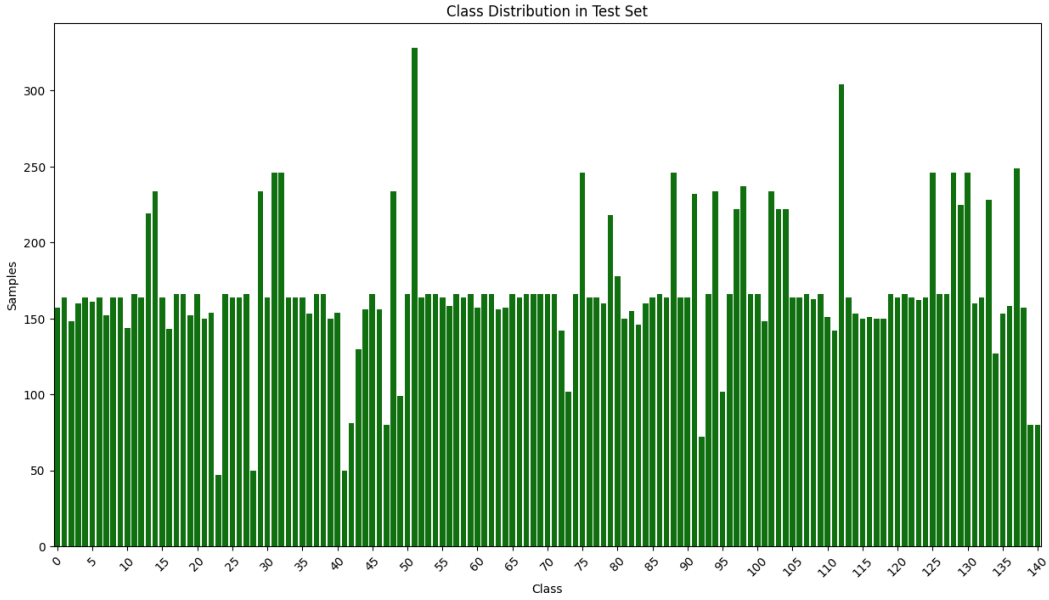

Figure 2: Distribuţia claselor în setul de testare.

2.2 Cerinţa 4.2.2 - Vizualizarea efectului de extragere a atributelor

Am afişat câte o imagine din primele 10 clase cu cele mai multe exemple, după aplicarea HoG, care ajută la capturarea trăsăturilor distinctive ale fiecărei clase, în speranţa că se vor evidenţia, în primul rând, texturile specifice fiecărui fruct, şi în al doilea rând conturul fructelor (deşi cele mai multe clase vor avea conturul oval).

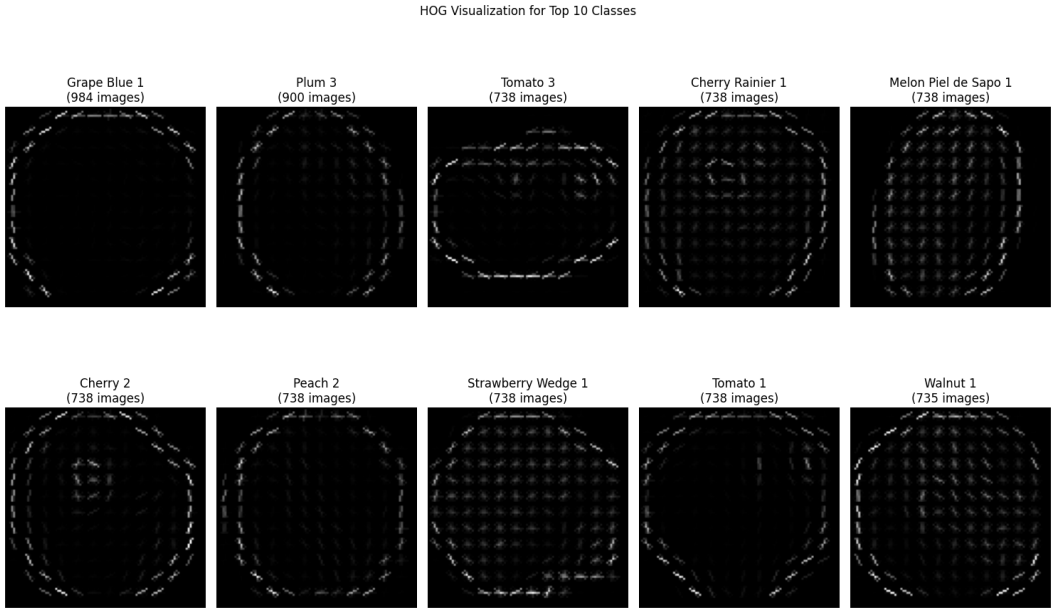

Figure 3: HoG aplicat pe primele 10 clase cu cele mai multe exemple.

Pentru PCA, mai întâi am calculat numărul de componente necesare pentru a păstra 95% varianţă a setului de date (am ales valoarea de 95% drept un compromis între acurateţea de clasificare şi cantitatea de date de procesat de modelele de învăţare automată, implicit şi timpul de rulare), apoi am afişat câte o imagine din primele 10 clase cu cele mai multe exemple, înainte şi după reconstrucţia imaginilor.

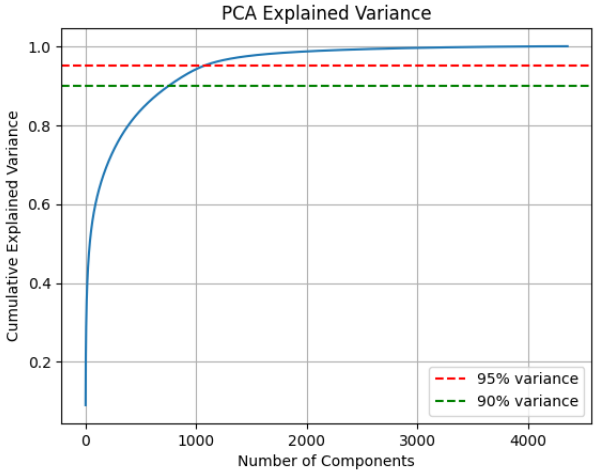

Figure 4: Număr de componente vs. varianţă.

```
Number of components for 95% variance: 1067
Number of components for 90% variance: 754
```

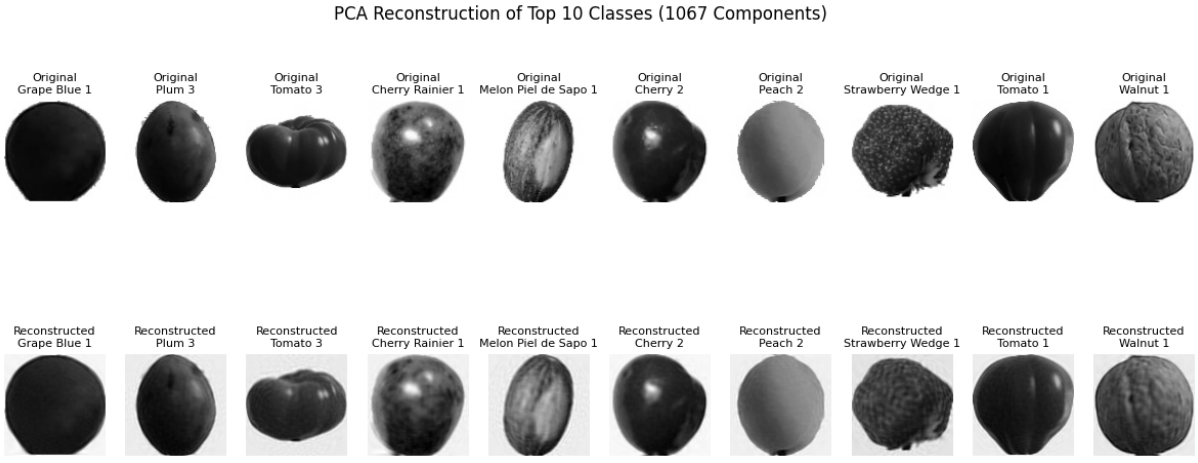

Figure 5: Imagini reconstruite cu 1067 de componente.

Utilitatea în a folosi PCA este, desigur, reducerea numărului de atribute, ceea ce a scăzut complexitatea computațională a antrenării modelelor de învățare automată, făcând procesul mai rapid și mai eficient.

3 Cerinţa 4.3 - Standardizarea şi selecţia atributelor

Pentru preprocesarea datelor, am utilizat StandardScaler pentru a uniformiza valorile numerice ale atributelor. Această etapă este importantă deoarece datele pot conține atribute cu scări de valori diferite, ceea ce poate afecta performanța clasificatorilor precum SVM.

Pentru reducerea dimensionalității setului de date am aplicat SelectPercentile, care selectează cele mai relevante atribute pe baza scorului ANOVA. Am ales să selectez primele 50% cele mai importante atribute, care au cea mai mare influență asupra predicţiei, eliminând pe cele redundante sau irelevante, de unde am ajuns la 533 de atribute cu care se face clasificarea. De asemenea, am afişat şi primele 20 atribute cu cea mai mare relevanţă pentru clasificare, în funcţie de scorul ANOVA:

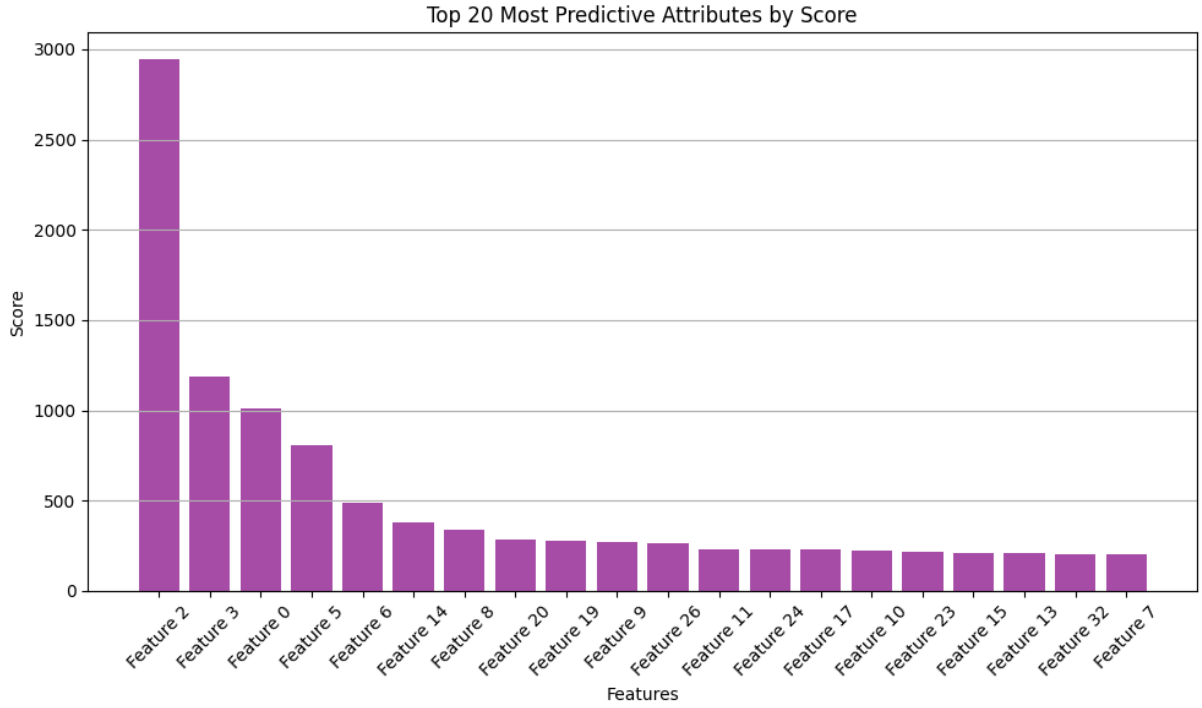

Figure 6: Primele 20 cele mai predictive atribute.

4 Cerinţa 4.4 - Modele de învăţare automată

Pentru antrenarea celor 4 modele de învăţare automată, am folosit setul de date cu primele 50% atributele selectate pe baza scorului ANOVA, de la pasul anterior. Pentru căutarea hiperparametrilor am folosit RandomizedSearchCV. Deoarece setul de date conţine foarte multe clase, pot apărea probleme la vizualizarea matricelor de confuzie sau a raportului de clasificare, aşa că am decis să afişez metrici doar pentru primele 20 cele mai slabe clase, în funcţie de scorul F1. Acurateţea afisată este pentru tot setul de date. La finalul documentului am ataşat şi matricele de confuzie pentru fiecare algoritm, dar cu clasele grupate (de exemplu: "Apple 1", "Apple 2" şi "Apple 3" devin o singură clasă, "Apple", etc.)

4.1 Regresie Logistică

```
C = 10
multi_class = "multinomial"
solver = "lbfgs"
```

O valoare mare a lui C înseamnă o regularizare mai slabă, ceea ce permite modelului să se potrivească mai bine cu datele de antrenare, dar poate duce la overfitting. Hiperparametrul "multi_class" este setat să folosească funcţia softmax, care calculează probabilitățile pentru toate clasele simultan și apoi alege clasa cu probabilitatea cea mai mare, "solver" este algoritmul utilizat pentru a minimiza funcția de cost, iar alegerea solver-ului influențează viteza de convergență și precizia modelului.

Table 1: Clasificare regresie logistică (cele mai slabe 20 de clase după scorul F1)

Class	Precision	Recall	F1-Score	Support
Potato Red 1	0.32	0.50	0.39	150
Pear Monster 1	0.48	0.35	0.40	166
Nectarine 1	0.39	0.46	0.42	164
Onion White 1	0.42	0.54	0.47	146
Pear 2	0.59	0.41	0.48	232
Potato White 1	0.41	0.60	0.48	150
Beetroot 1	0.51	0.47	0.49	150
Corn 1	0.71	0.40	0.51	150
Plum 1	0.57	0.51	0.54	151
Potato Sweet 1	0.53	0.56	0.54	150
Potato Red Washed 1	0.58	0.53	0.55	151
Eggplant 1	0.84	0.44	0.58	156
Apple Red Yellow 1	0.67	0.55	0.60	164
Pepper Orange 1	0.85	0.47	0.60	234
Chestnut 1	0.66	0.59	0.63	153
Banana Red 1	0.88	0.49	0.63	166
Onion Red Peeled 1	0.61	0.66	0.64	155
Corn Husk 1	0.75	0.57	0.65	154
Apple Red 1	0.68	0.62	0.65	164
Banana Lady Finger 1	0.60	0.73	0.66	152

Table 2: Metrici pentru regresia logistică (întreg setul de date)

Metric				Support
Accuracy			0.83	23619
Macro Avg	0.84	0.83	0.83	23619
Weighted Avg	0.84	0.83	0.83	23619

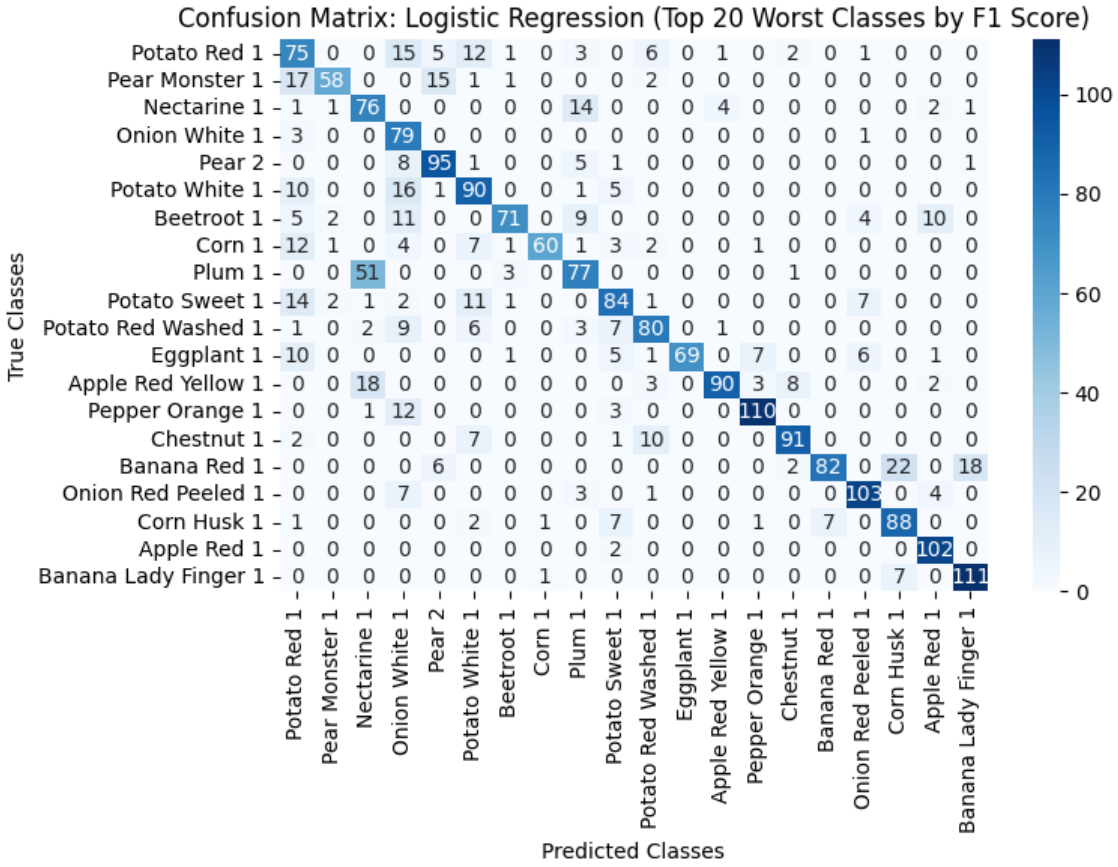

Figure 7: Matricea de confuzie pentru regresia logistică (primele 20 cele mai slabe clase în funcţie de scorul F1)

4.2 SVM

```
C = 0.1
kernel = "linear"
gamma = "scale"
class_weight = "balanced"
```

La fel ca la regresia logistică, factorul de regularizare C înseamnă o penalizare mai mică pentru erorile de clasificare, ceea ce permite modelului să învețe mai bine datele de antrenare, dar poate duce la overfitting, iar hiperparametrul "kernel" este setat pe "linear", ceea ce indică faptul că modelul poate separa clasele printr-un hiperplan. Parametrul gamma asigură că influența fiecărei caracteristici asupra deciziei este uniformă, iar "class_weight" ajustează automat ponderea fiecărei clase în funcție de distribuția claselor din setul de date şi ajută modelul să evite favorizarea claselor dominante.

Table 3: Clasificare SVM (cele mai slabe 20 de clase după scorul F1)

Class	Precision	Recall	F1-Score	Support
Nectarine 1	0.43	0.48	0.45	164
Potato Red 1	0.46	0.55	0.50	150
Corn 1	0.76	0.42	0.54	150
Pear Monster 1	0.62	0.50	0.55	166
Pear 2	0.69	0.47	0.56	232
Beetroot 1	0.61	0.51	0.56	150
Onion White 1	0.48	0.68	0.57	146
Potato White 1	0.50	0.68	0.58	150
Potato Sweet 1	0.64	0.59	0.61	150
Potato Red Washed 1	0.69	0.58	0.63	151
Plum 1	0.69	0.63	0.66	151
Banana Red 1	0.94	0.51	0.66	166
Corn Husk 1	0.75	0.61	0.67	154
Pepper Orange 1	0.93	0.54	0.68	234
Banana Lady Finger 1	0.69	0.72	0.70	152
Apple Red 2	0.67	0.76	0.71	164
Onion Red Peeled 1	0.75	0.73	0.74	155
Maracuja 1	0.72	0.77	0.74	166
Apple Red Yellow 1	0.75	0.73	0.74	164
Pear Stone 1	0.65	0.88	0.75	237

Table 4: Metrici pentru SVM (întreg setul de date)

Metric				Support
Accuracy			**0.88**	23619
Macro Avg	0.89	0.88	0.88	23619
Weighted Avg	0.89	0.88	0.88	23619

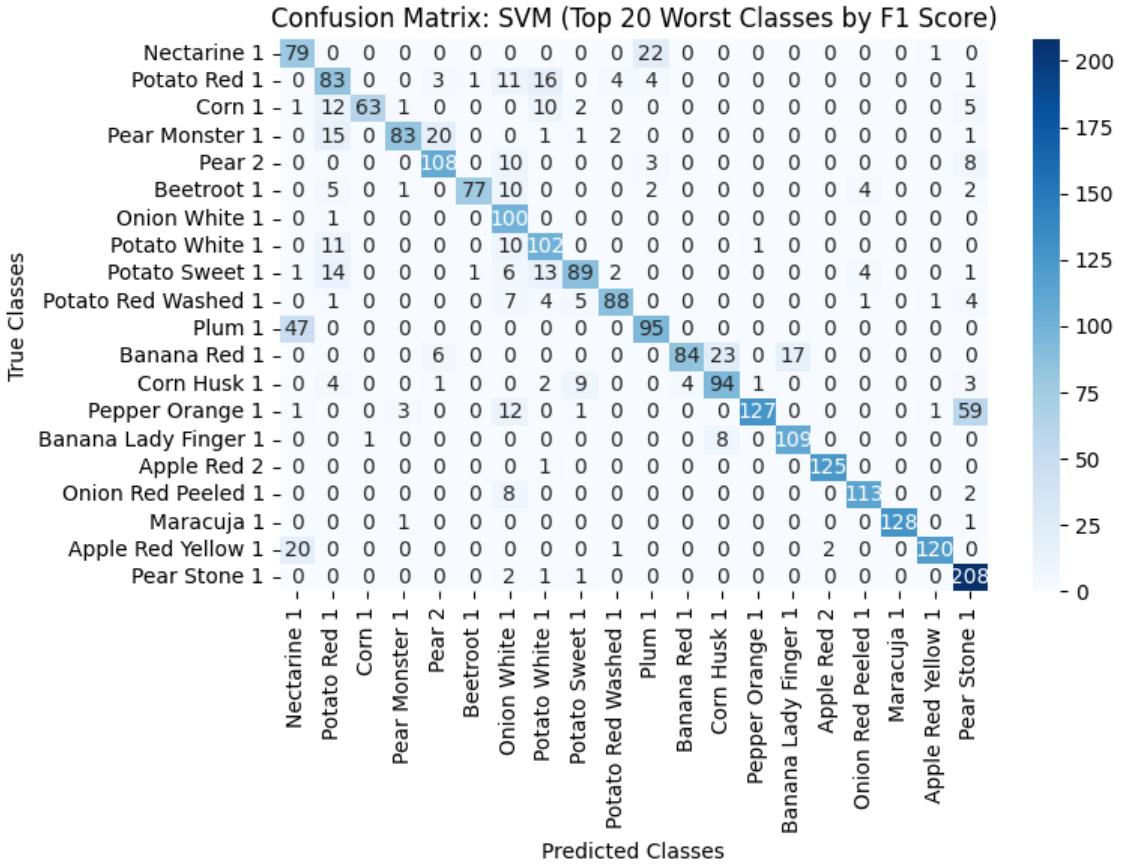

Figure 8: Matricea de confuzie pentru SVM (primele 20 cele mai slabe clase în funcţie de scorul F1)

4.3 Random Forest

```
n_estimators = 200
max_features = "sqrt"
max_depth = 60
```

Hiperparametrul "n_estimators" definește numărul de arbori în pădurea aleatoare, o valoare mai mare duce la un model mai robust, dar cu costuri computaționale mai mari. max_features = "sqrt" indică faptul că pentru fiecare arbore, se va alege un subset aleatoriu de atribute (rădăcina pătrată din numărul total de atribute), ceea ce poate preveni overfitting-ul, reducând corelaţia dintre arbori. Parametrul "max_depth" controlează adâncimea maximă a fiecărui arbore, însă o valoare prea mare poate duce la overfitting.

Table 5: Clasificare Random Forest (cele mai slabe 20 de clase după scorul F1)

Class	Precision	Recall	F1-Score	Support
Potato Red 1	0.44	0.34	0.38	150
Pear 2	0.47	0.36	0.41	232
Onion White 1	0.32	0.62	0.42	146
Nectarine 1	0.50	0.38	0.43	164
Beetroot 1	0.52	0.41	0.46	150
Potato Sweet 1	0.52	0.42	0.46	150
Potato Red Washed 1	0.72	0.35	0.47	151
Corn 1	0.61	0.39	0.48	150
Potato White 1	0.56	0.43	0.48	150
Onion Red Peeled 1	0.67	0.50	0.57	155
Pear Monster 1	0.84	0.46	0.59	166
Apple Crimson Snow 1	0.45	0.90	0.60	148
Apple Red 1	0.78	0.49	0.60	164
Maracuja 1	0.57	0.67	0.62	166
Plum 1	0.72	0.56	0.63	151
Apple Golden 3	0.62	0.67	0.64	161
Banana Red 1	0.95	0.49	0.65	166
Walnut 1	0.48	0.99	0.65	249
Corn Husk 1	0.74	0.58	0.65	154
Pepper Orange 1	0.78	0.58	0.67	234

Table 6: Metrici pentru Random Forest (întreg setul de date)

Metric				Support
Accuracy			0.81	23619
Macro Avg	0.83	0.81	0.81	23619
Weighted Avg	0.83	0.81	0.81	23619

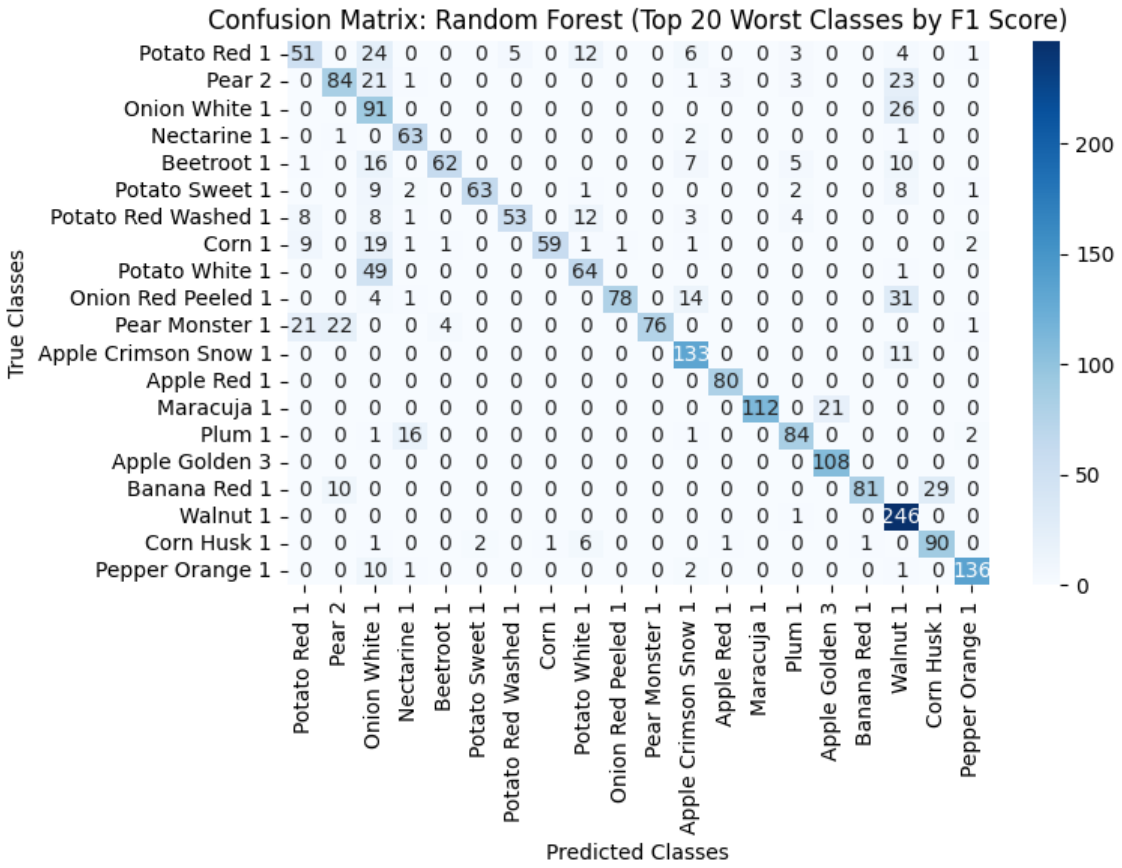

Figure 9: Matricea de confuzie pentru Random Forest (primele 20 cele mai slabe clase în funcţie de scorul F1)

4.4 Gradient Boosting

```
n_estimators = 200
max_depth = 6
learning_rate = 0.1
eval_metric = "logloss"
tree_method = "hist"
```

Asemănător ca la Random Forest, "n_estimators" şi "max_depth" controlează numărul de arbori şi respectiv adâncimea maximă a unui arbore, iar "learning_rate" controlează cât de mult contribuie fiecare arbore la ajustarea predicțiilor modelului şi impactează timpul de convergenţă al modelului. Logloss măsoară diferența între probabilitățile prezise de model și etichetele reale şi penalizează mai sever predicțiile greșite care sunt făcute cu încredere ridicată, iar Histogram-based tree method este un algoritm optimizat pentru construirea arborilor care utilizează histograme pentru a reduce timpul de calcul.

Table 7: Clasificare Gradient Boosting (cele mai slabe 20 de clase după scorul F1)

Class	Precision	Recall	F1-Score	Support
Nectarine 1	0.25	0.31	0.28	164
Potato Red 1	0.36	0.37	0.37	150
Beetroot 1	0.40	0.36	0.38	150
Onion White 1	0.35	0.51	0.42	146
Plum 1	0.39	0.46	0.42	151
Corn 1	0.58	0.35	0.43	150
Potato Sweet 1	0.54	0.38	0.45	150
Pear 2	0.54	0.41	0.46	232
Potato White 1	0.43	0.54	0.48	150
Potato Red Washed 1	0.68	0.38	0.49	151
Eggplant 1	0.63	0.41	0.50	156
Nectarine Flat 1	0.52	0.59	0.55	160
Apple Red 1	0.74	0.46	0.57	164
Apple Red Yellow 1	0.57	0.58	0.57	164
Cherry Rainier 1	0.62	0.55	0.58	246
Banana Red 1	0.77	0.47	0.58	166
Banana Lady Finger 1	0.52	0.68	0.59	152
Onion Red Peeled 1	0.73	0.50	0.60	155
Grape Pink 1	0.66	0.56	0.61	164
Corn Husk 1	0.76	0.51	0.61	154

Table 8: Metrici pentru Gradient Boosting (întreg setul de date)

Metric				Support
Accuracy			0.78	23619
Macro Avg	0.79	0.78	0.78	23619
Weighted Avg	0.79	0.78	0.77	23619

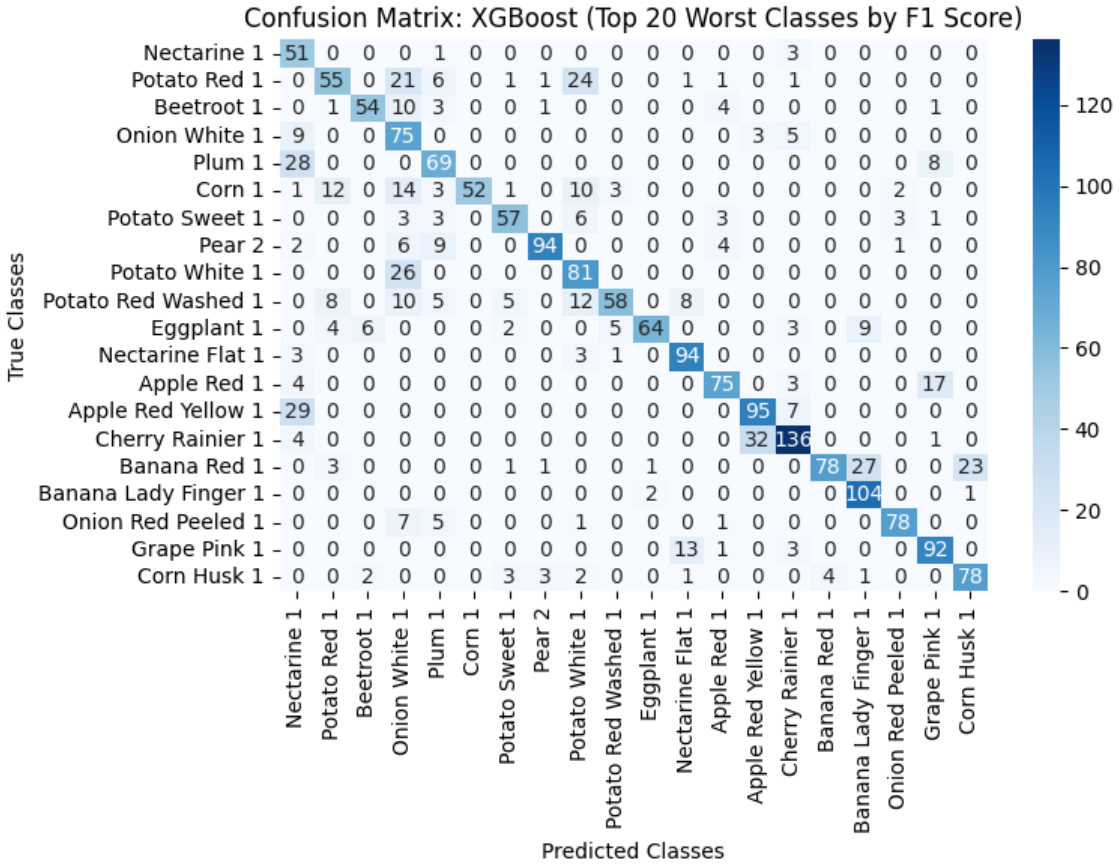

Figure 10: Matricea de confuzie pentru Gradient Boosting (primele 20 cele mai slabe clase în funcţie de scorul F1)

Aşadar, după cum se poate observa, aceste modelele de învăţare automată reuşesc să clasifice destul de bine imaginile, pentru unele clase chiar clasificându-le pe toate corect. De asemenea, clasificarea greşită a unor imagini se poate explica prin faptul că pur şi simplu structura fructele (formă, textură, culoare) seamănă între ele îndeajuns de mult încât clasificatorul să nu poată reuşi să le diferenţieze (de exemplu: "Apple Red Yellow 1" cu "Nectarine 1" - ambele fructe au aceeaşi formă, aceleaşi culori şi o textură asemanătoare; "Cherry Rainier 1" cu "Apple Red Yellow 1" - care iar seamănă foarte mult ca trasături între ele)

4.5 Afişări suplimentare

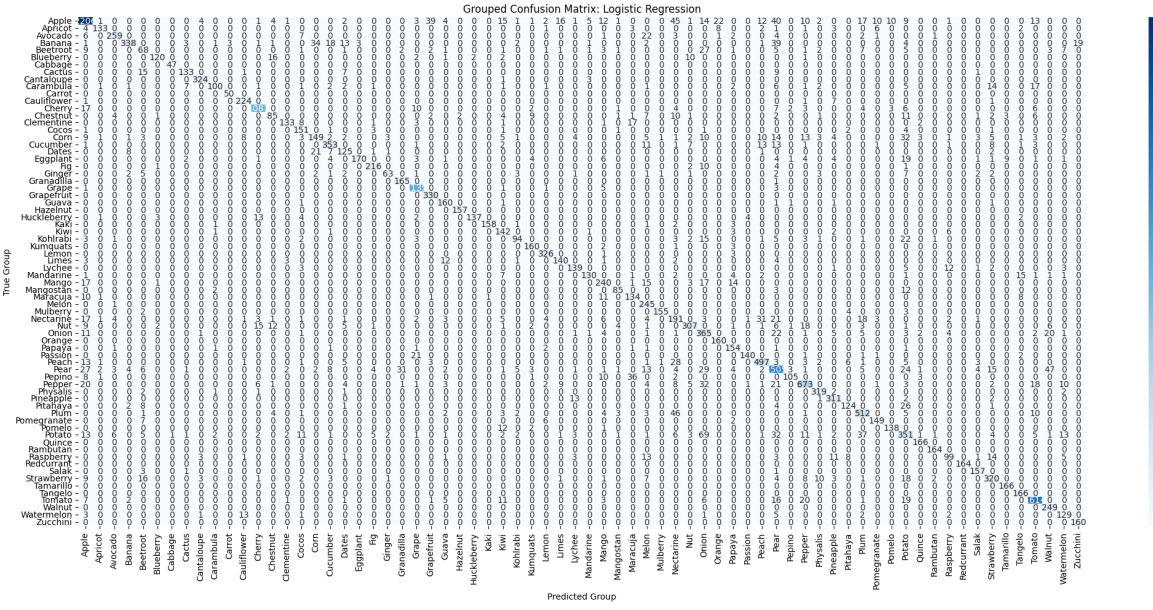

Figure 11: Matricea de confuzie cu clasele grupate pentru regresia logistică.

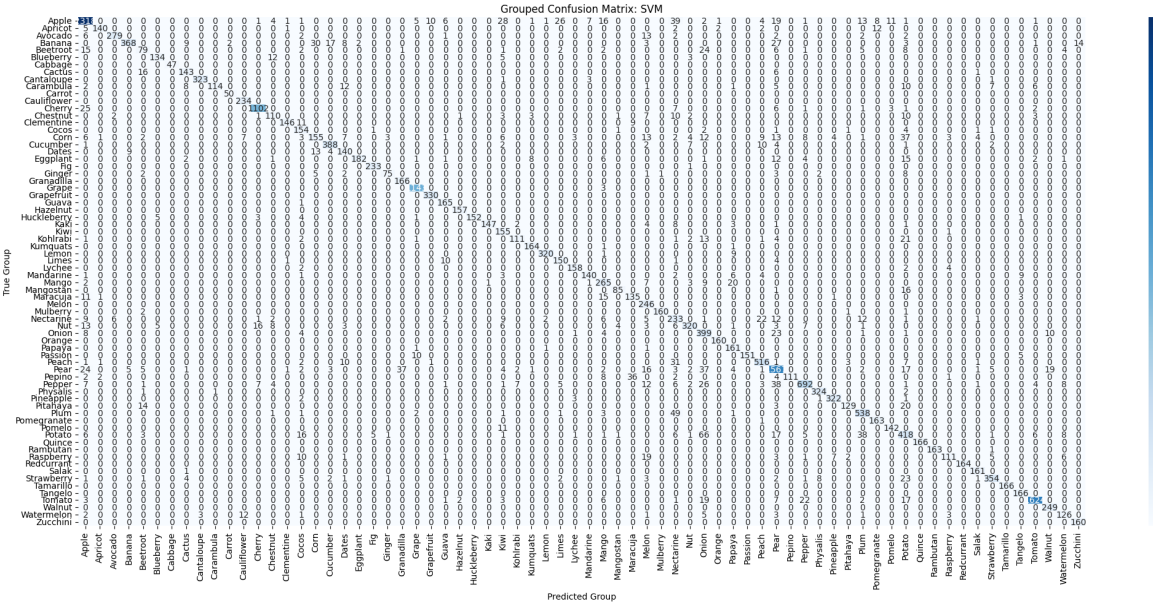

Figure 12: Matricea de confuzie cu clasele grupate pentru SVM.

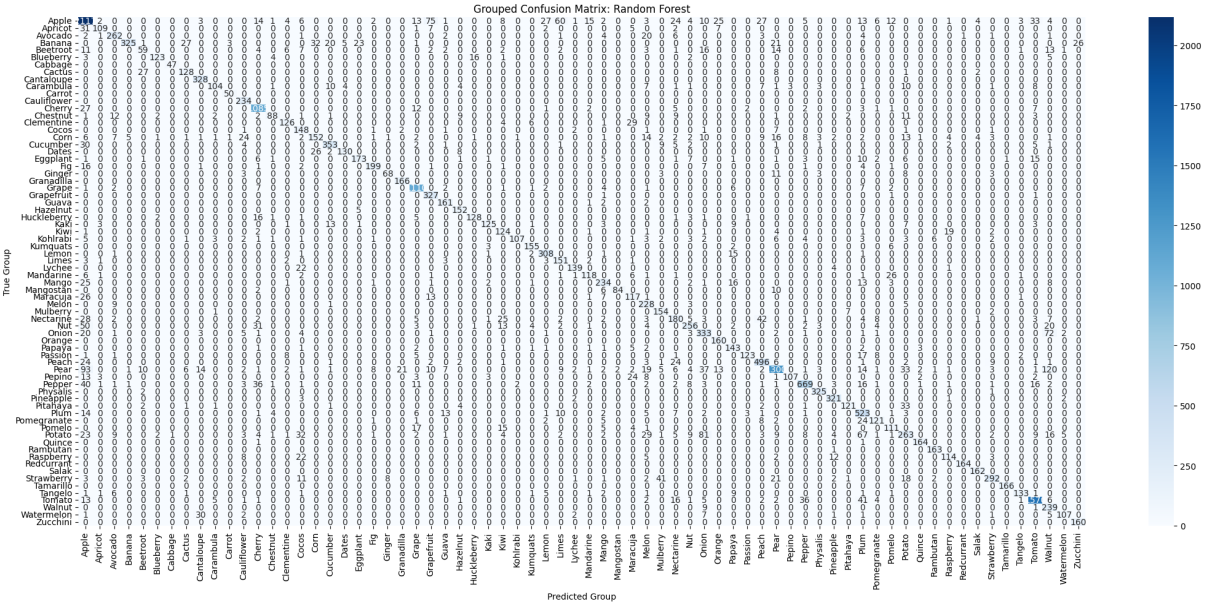

Figure 13: Matricea de confuzie cu clasele grupate pentru Random Forest.

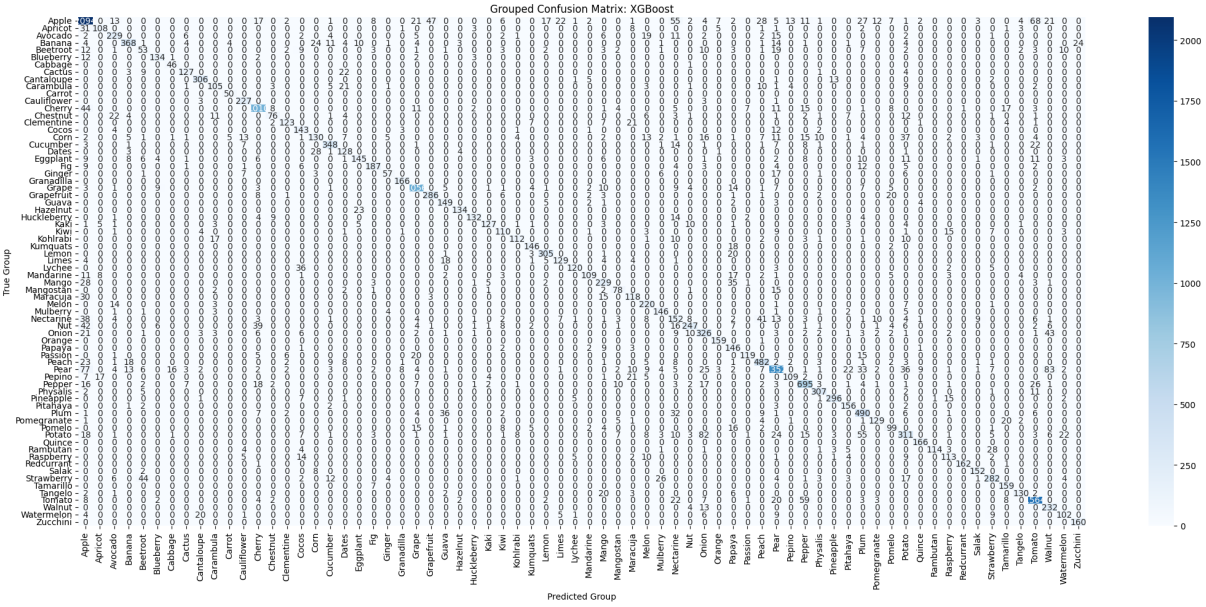

Figure 14: Matricea de confuzie cu clasele grupate pentru Gradient Boosting.