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:

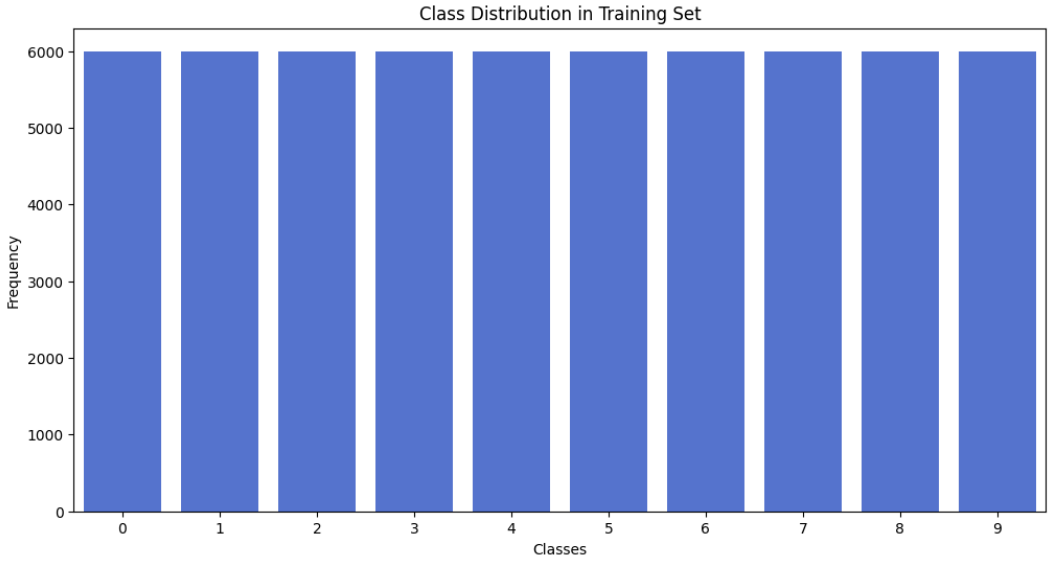

Figure 1: Distribuţia claselor în setul de antrenare.

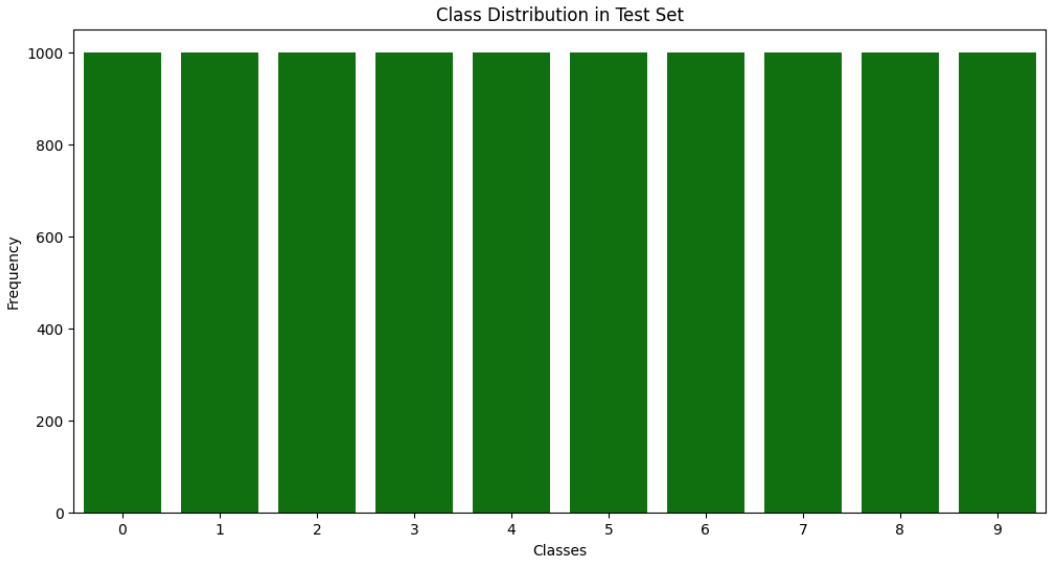

Figure 2: Distribuţia claselor în setul de testare.

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.

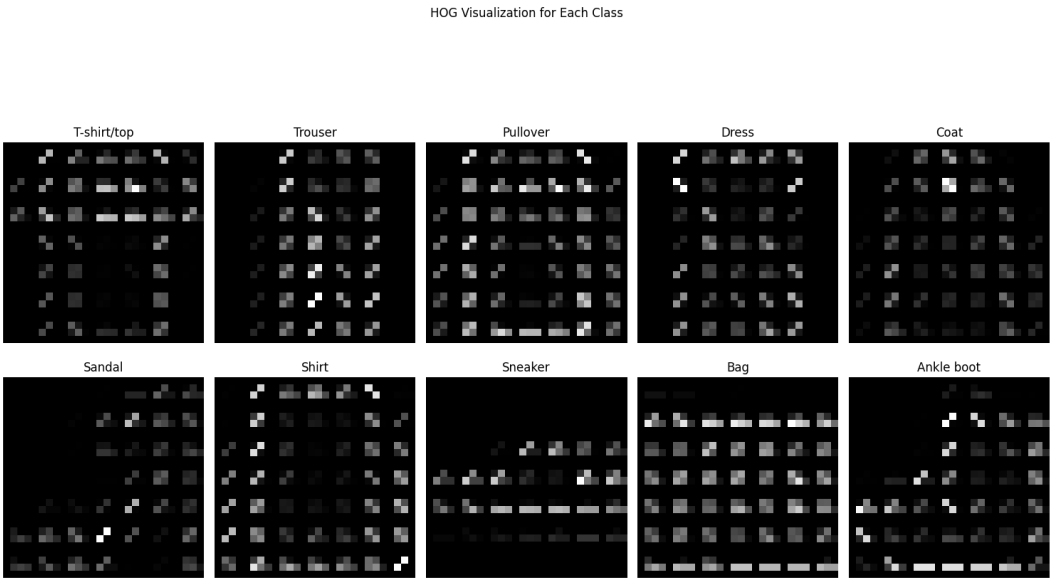

Figure 3: HoG aplicat pe câte o imagine din fiecare clasă.

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.

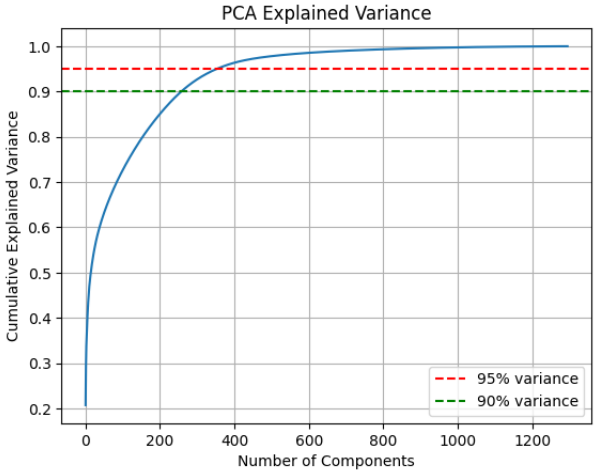

Figure 4: Număr de componente vs. varianţă.

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

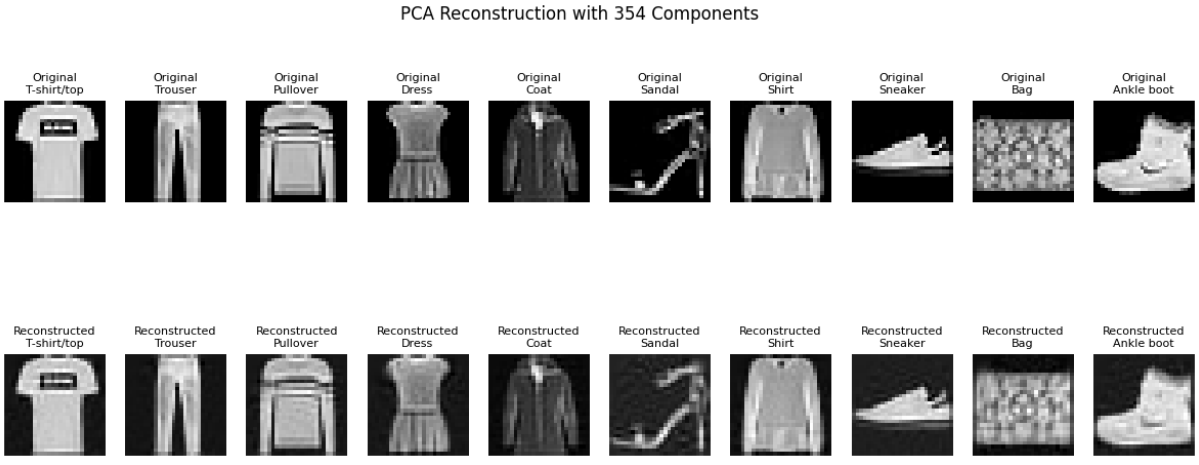

Figure 5: Imagini reconstruite cu 354 de componente.

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:

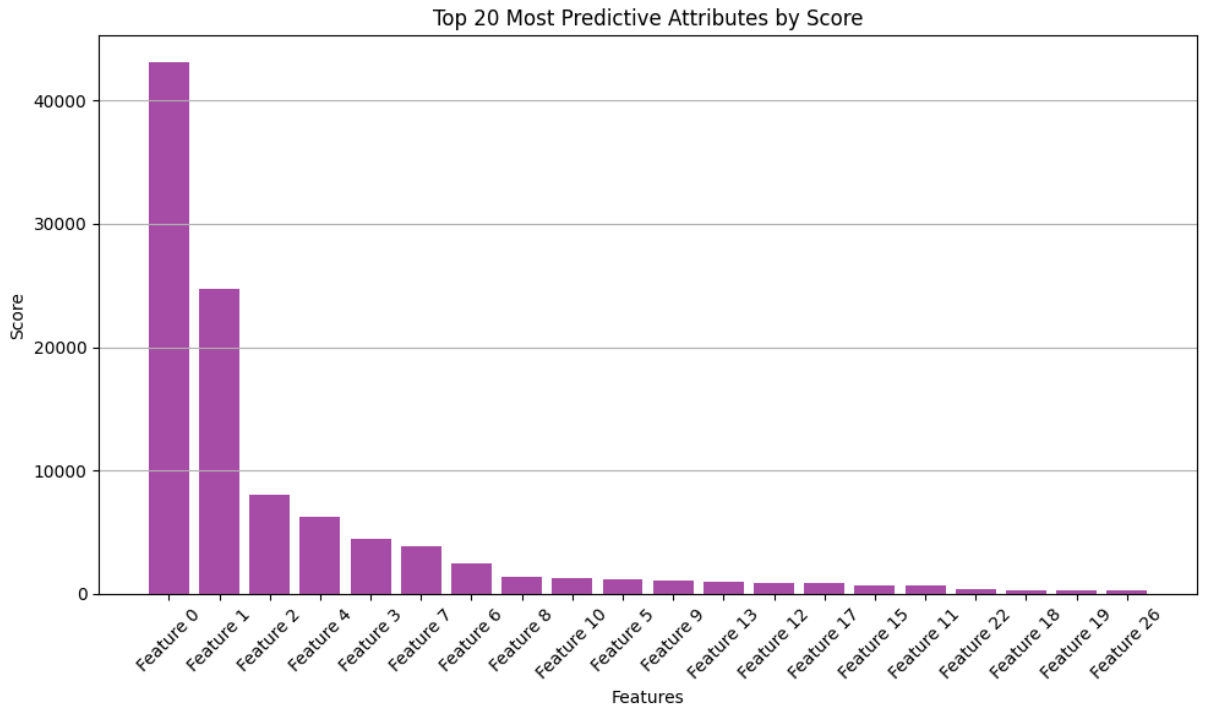

Figure 6: Primele 20 cele mai predictive atribute.

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

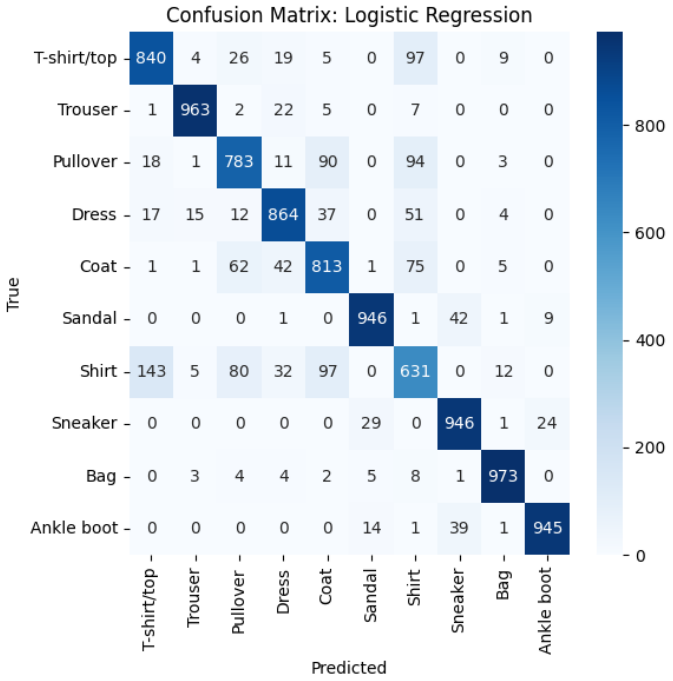

Figure 7: Matricea de confuzie pentru regresia logistică.

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

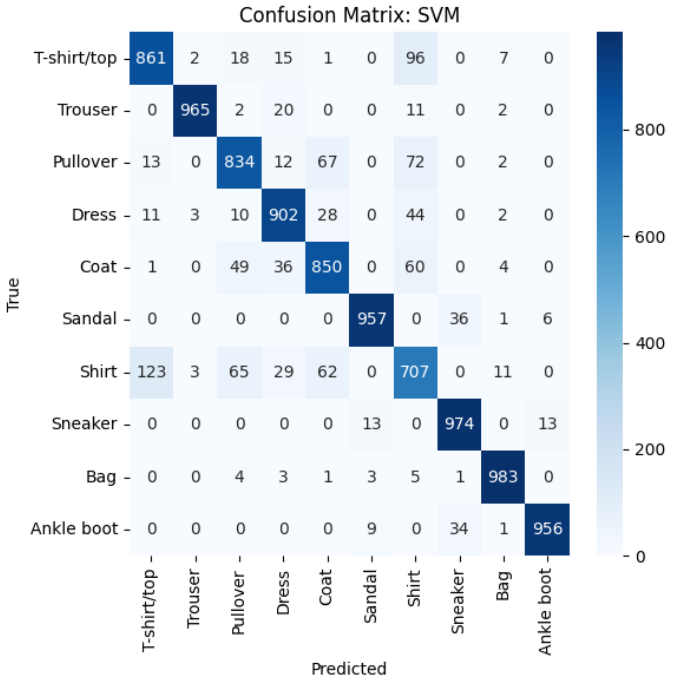

Figure 8: Matricea de confuzie pentru SVM.

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

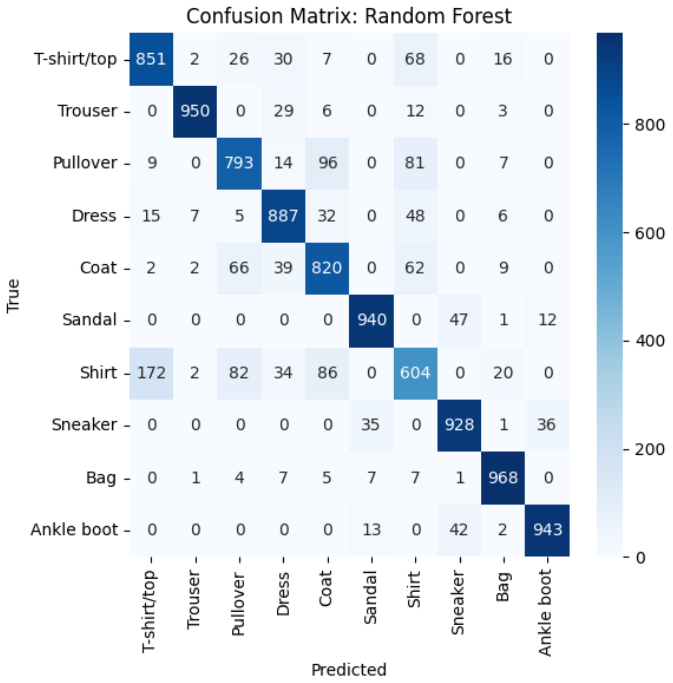

Figure 9: Matricea de confuzie pentru Random Forest.

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

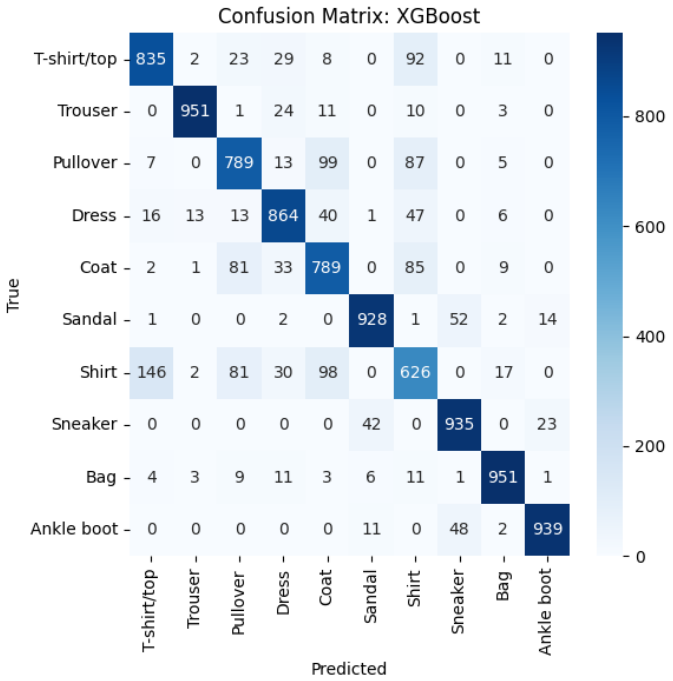

Figure 10: Matricea de confuzie pentru Gradient Boosting.

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%.