# Clasificare de imagini pe baza algoritmilor de invatare automata

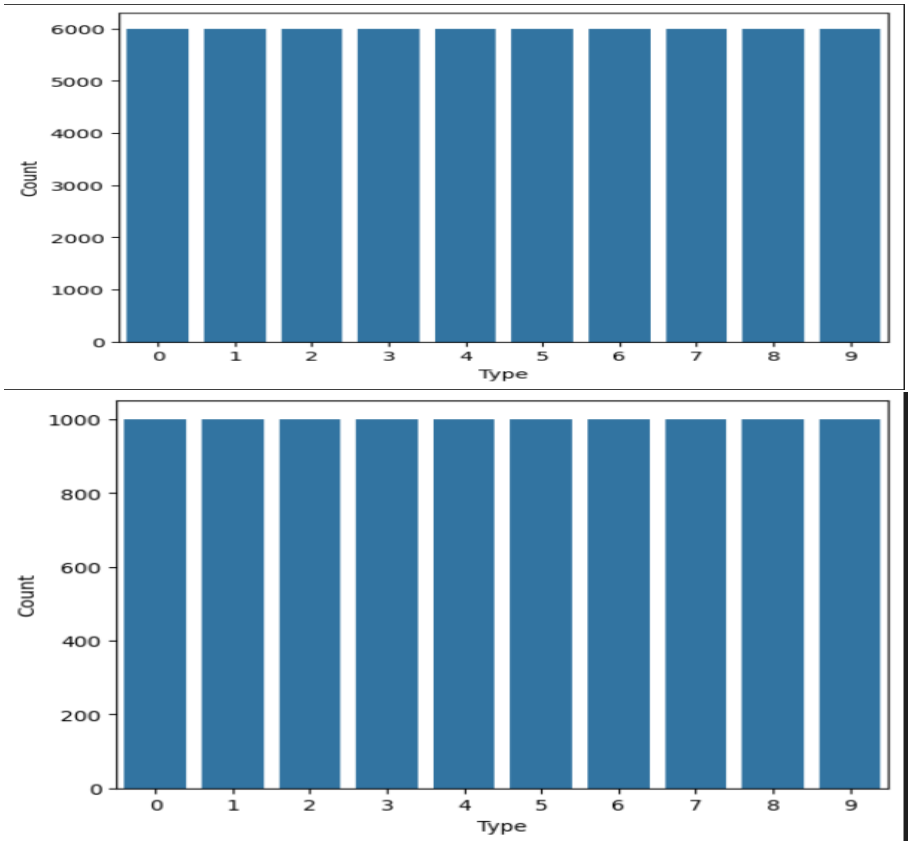
*Stan Petrișor Cătălin, 342C3*

## 1. *Fashion-MNIST*

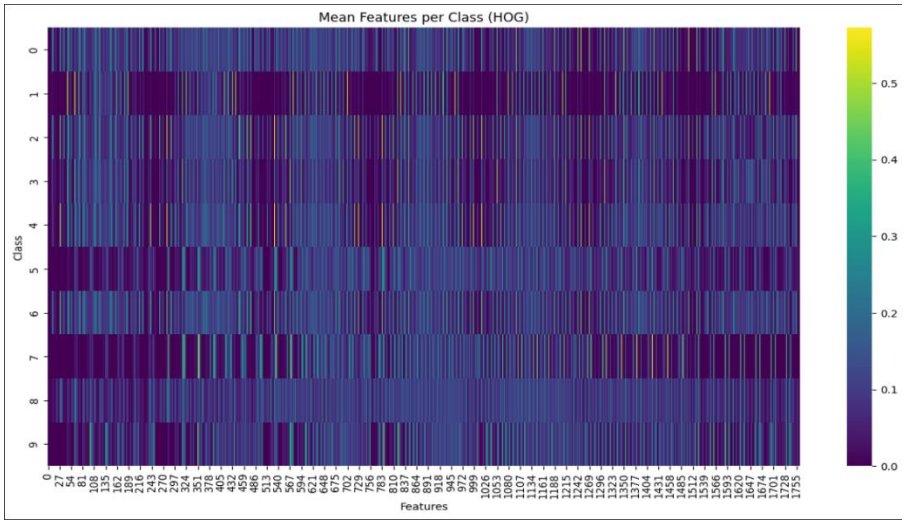
### *EDA*

Metode alese pentru extragerea de attribute din imagini:

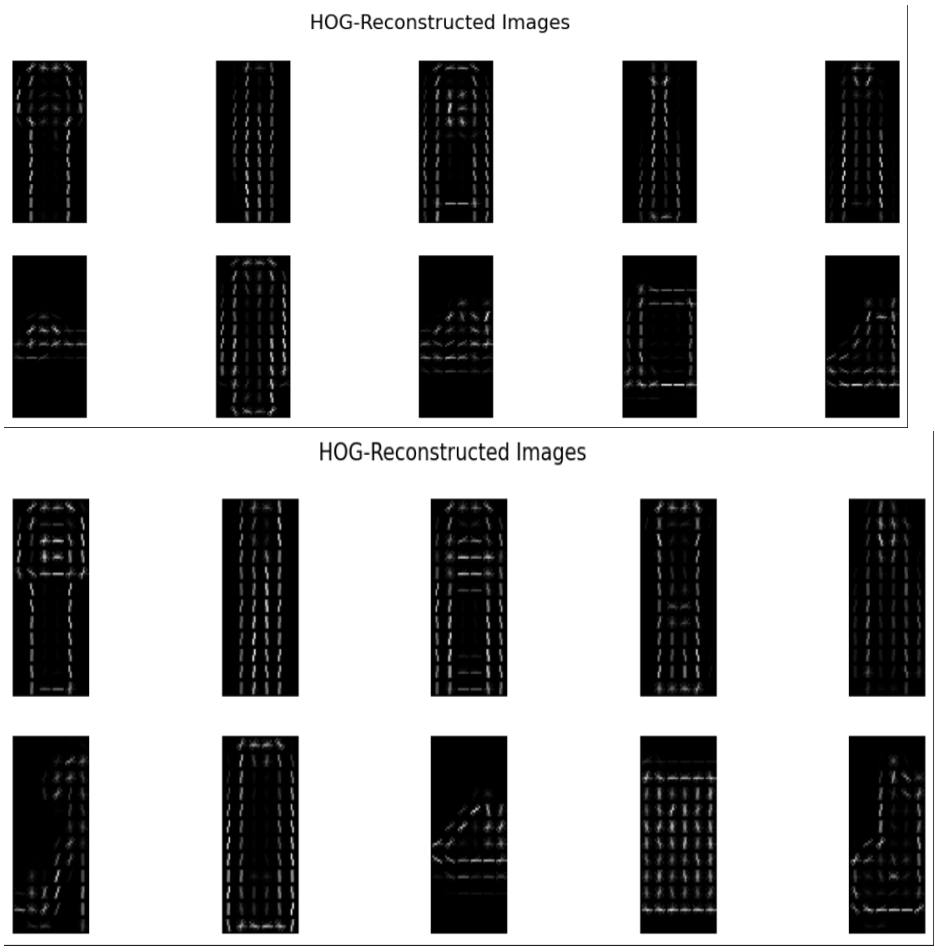
a. HOG(Histogram of Gradients)
   i. Alegere facuta deoarece analiza hainelor nu tine cont de culoarea acestora ci doar de forma lor, **HOG** fiind un bun candidat pentru determinarea orientarii varfurilor obiectului(ce determina la randul lor forma acestuia).
   ii. Vizualizarile si statisticile cerute atat pe setul de antrenare cat si pe cel de testare:



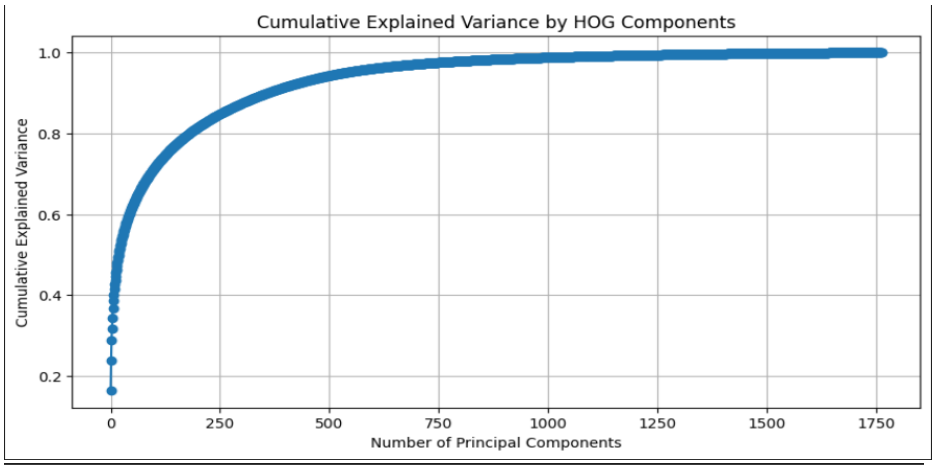

   iii. Putem observa un echilibru al claselor atat pe setul de antrenare cat si pe cel de testare, acestea fiind distribuite in mod egal.

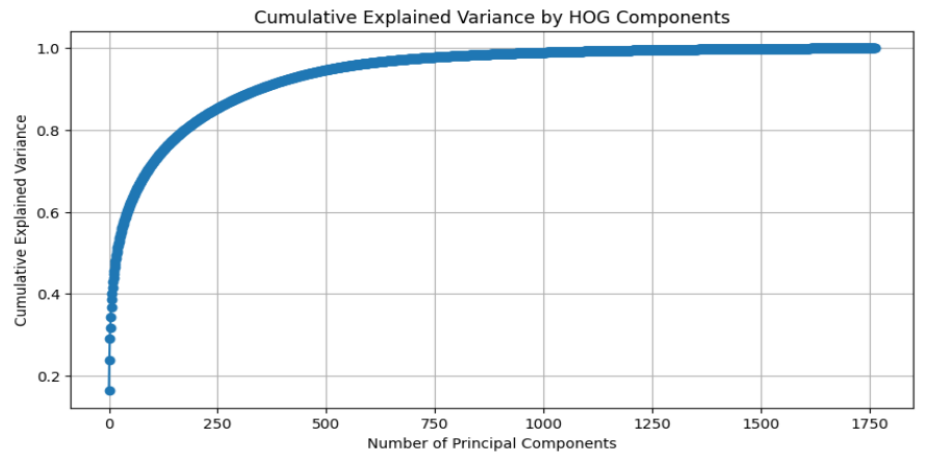




iv. Culoarea mov inchisa din heatmap-urile de mai sus(pentru fiecare clasa in parte) reprezinta atributele relevante ale unei imagini, ce dau de fapt forma si orientarea acestuia, pe cand culori foarte deschise de albastru(sau cele de galben) reprezinta atributele mai putin importante din imagine.




v. Mai sus se pot observa imaginile reconstruite. Acestea sunt de conturate datorita numarului mare de atribute obtinute in urma extragerii(am dat “resize” la imagini facandu-le de 64x64).

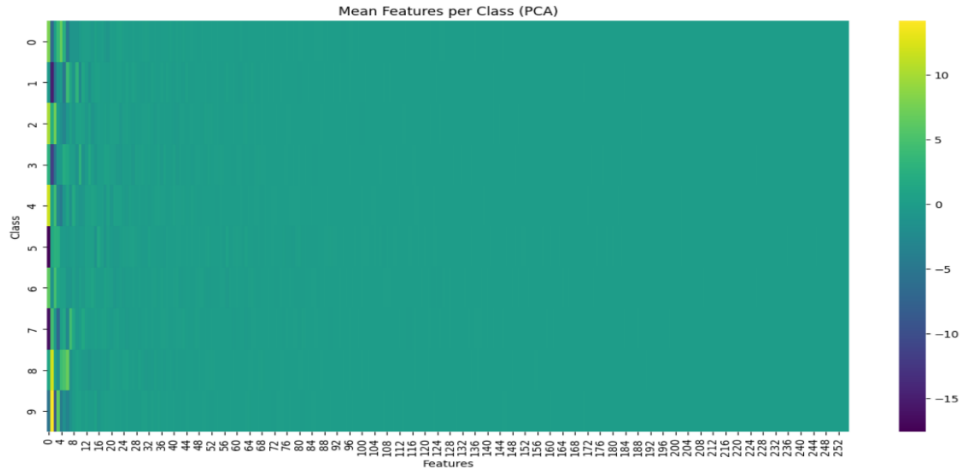


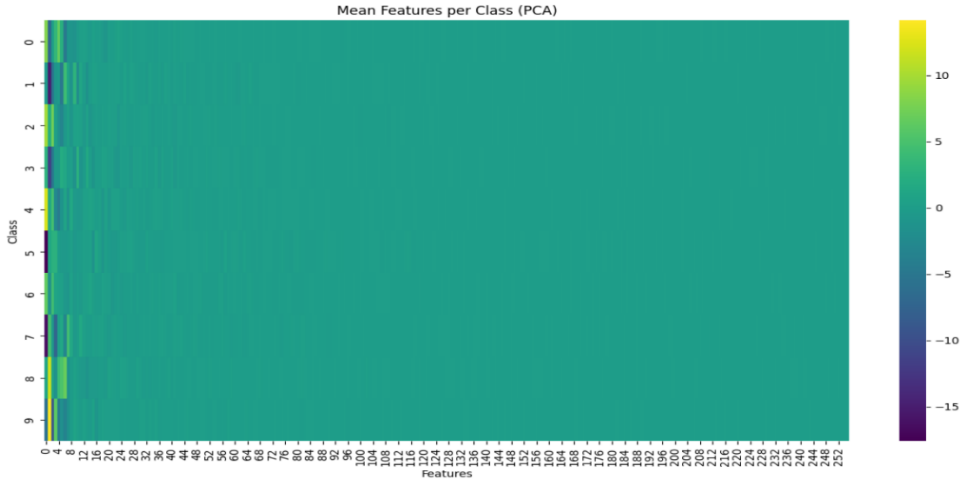




vi. In aceste 2 grafice am analizat varianta cumulativa per numarul de atribute ale imaginii. Initial se observa o crestere rapida a curbei(indicand faptul ca putine atribute dau cea mai mare parte din varianta, adica putine atribute sunt relevante pentru imagine). Dupa ~500 de componente observam contributia din ce in cem ai mica a atributelor adaugate, acestea nemaifiind intr-atat de relevante.

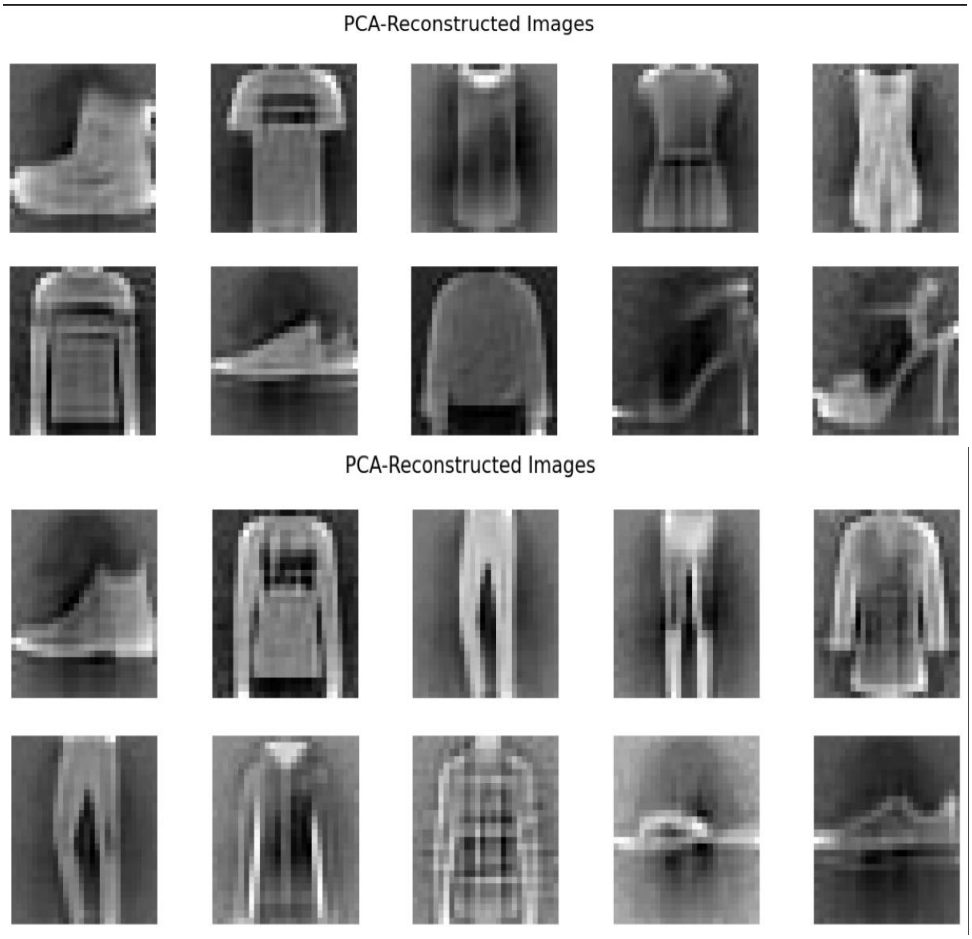
b. PCA(Principal Components Analysis)
   i. Alegere facuta din cauza simplitatii metodei, alegand sa extrag doar componentele care retin 95% din varianta.

ii. Vizualizarile si statisticile cerute atat pe setul de antrenare cat si pe cel de testare:

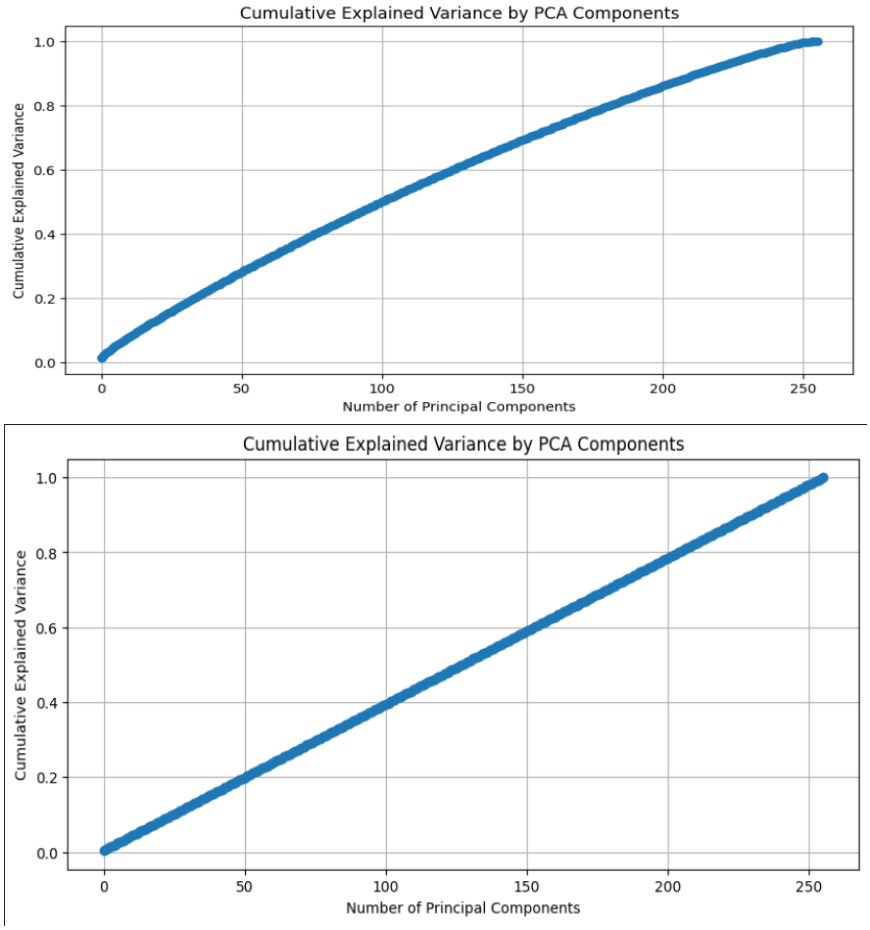


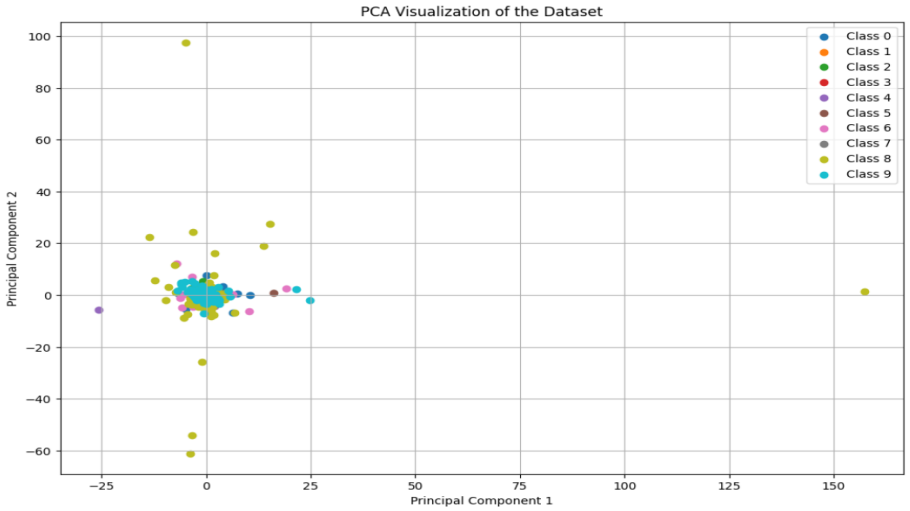

iii. Exact ca la **HOG,** putem observa si aici o importanta mult mai mare a atributelor extrase. Acest heatmap arata relevanta atributelor extrase cu **PCA,** o proportie semnificativa dintre ele fiind importante pentru descrierea imaginilor.

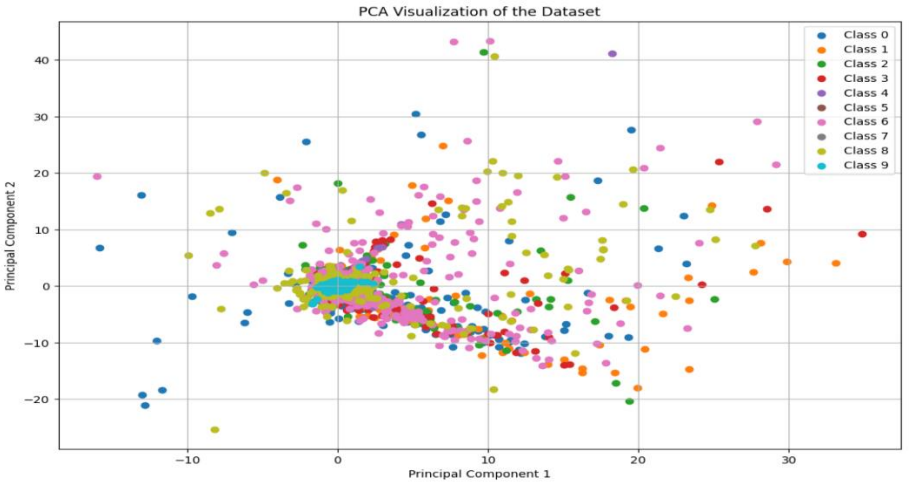



iv. Aici putem observa imaginile reconstruite folosind cele 256 de componente extrase.




v. Se poate explica in aceste ilustratii o crestere mai lenta a variantiei cumulative in raport cu numarul de atribute extrase, deoarece cum am observant si in heatmap-ul de mai sus, toate componentele sunt relevante imaginii.




vi. Aceste 2 grafice arata suprapunerea semnificativa a claselor, cee ace sugereaza ca acestea sunt greu de separate utilizandu-se doar un spatiu bidimensional(trebuia facuta o diagrama 3D aici pentru o vizualizare mai concisa).

## *Data Preprocessing*

Am ales pentru standardizarea datelor **StandardScaler**, iar pentru selectia atributelor am utilizat **SelectPercentile** cu parametrul **percentile** avand valoarea 10. Setul initial de date(cel obtinut in uma aplicarii metodelor de extractie prezentate anterior) difera de setul de date obtinut dupa selectie prin numarul de atribute alese, dar totusi se aseamana prin relevanta datelor ramase, **SelectPercentile** selectand cele mai puternice 10% atribute din cele initiale.

## *Models Training and Performance Counters*

Algoritmii de Invatare Automata utilizati au prezentat urmatoarele performante:

a. **Logistic Regression**
   1. Timp executie: 165.7 secunde
   2. Rezultate pentru combinatii de hiperparametrii:

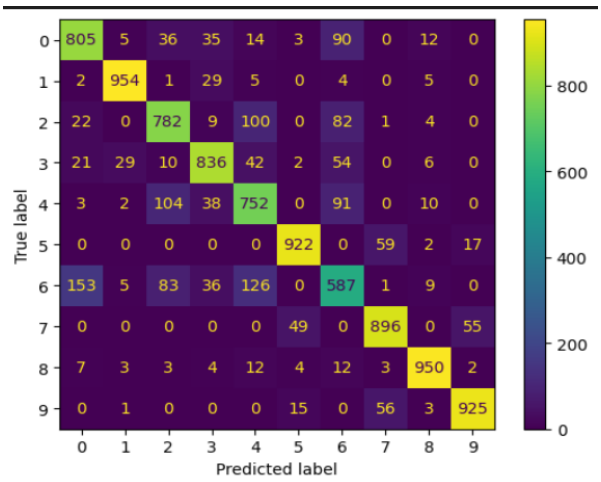
| param_C | param_multi_class | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|
| | | | | | | |
| 10 | ovr | 0.844 | **0.001** | 0.841 | 0.844 | 0.842 |
| 10 | multinomial | **0.845** | 0 | **0.844** | **0.845** | **0.844** |
| 20 | ovr | 0.844 | **0.001** | 0.841 | 0.844 | 0.842 |
| 20 | multinomial | **0.845** | **0.001** | 0.843 | **0.845** | **0.844** |
| 30 | ovr | 0.844 | **0.001** | 0.841 | 0.844 | 0.842 |
| 30 | multinomial | **0.845** | **0.001** | **0.844** | **0.845** | **0.844** |
| 40 | ovr | 0.844 | **0.001** | 0.841 | 0.844 | 0.842 |
| 40 | multinomial | **0.845** | **0.001** | 0.843 | **0.845** | **0.844** |
| 50 | ovr | 0.844 | **0.001** | 0.841 | 0.844 | 0.842 |
| 50 | multinomial | 0.844 | **0.001** | 0.843 | 0.844 | 0.843 |
| 60 | ovr | 0.844 | **0.001** | 0.841 | 0.844 | 0.842 |
| 60 | multinomial | **0.845** | **0.001** | **0.844** | **0.845** | **0.844** |
| 70 | ovr | 0.843 | **0.001** | 0.841 | 0.843 | 0.842 |
| 70 | multinomial | **0.845** | **0.001** | **0.844** | **0.845** | **0.844** |
| 80 | ovr | 0.843 | **0.001** | 0.841 | 0.843 | 0.842 |
| 80 | multinomial | **0.845** | **0.001** | **0.844** | **0.845** | **0.844** |
| 90 | ovr | 0.843 | **0.001** | 0.841 | 0.843 | 0.842 |
| 90 | multinomial | 0.844 | **0.001** | 0.843 | 0.844 | **0.844** |
| 100 | ovr | 0.843 | **0.001** | 0.841 | 0.843 | 0.841 |
| 100 | multinomial | **0.845** | **0.001** | **0.844** | **0.845** | **0.844** |

Observam ca toate combinatiile de forma (C, **multinomial**) obtin performante mai bune in clasificare comparative cu cele ce folosesc **ovr.**

3. Raport de clasificare si hiperparametrii optimi determinati:

```
✓ 2m 45.7s

Fitting 3 folds for each of 20 candidates, totalling 60 fits
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfr
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_4536\3624232938.py:76: SettingWithCopyWa
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  metrics_table.rename(columns={
Best hyperparameters: {'multi_class': 'multinomial', 'C': 80}
Best hyperparameters selected:
{'multi_class': 'multinomial', 'C': 80}

Accuracy: 0.8409
F1-score: 0.8409
Recall: 0.8409

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.81      0.80      1000
           1       0.95      0.95      0.95      1000
           2       0.77      0.78      0.77      1000
           3       0.85      0.84      0.84      1000
           4       0.72      0.75      0.73      1000
           5       0.93      0.92      0.92      1000
           6       0.64      0.59      0.61      1000
           7       0.88      0.90      0.89      1000
           8       0.95      0.95      0.95      1000
           9       0.93      0.93      0.93      1000

    accuracy                           0.84     10000
   macro avg       0.84      0.84      0.84     10000
weighted avg       0.84      0.84      0.84     10000
```

4. Matrice de confuzie obtinuta:




Pe baza raportului de clasificare si a matricei de confuzie obtinuta se pot observa clasele cu cele mai bune predictii: **Trouser**(1), **Sandal**(5), **Bag(8), Ankle Boot**(9). Celelalte clase prezinta o acuratete la fel de buna aproape, exceptie facand clasa **Shirt**(6) unde se poate observa confuzia facuta cu clasele **T-shirt**, **Pullover** si **Coat**, acestea avand valori similare atributelor dupa cum se poate observa in urma extragerii atributelor cu **HOG**.

**b. SVM**

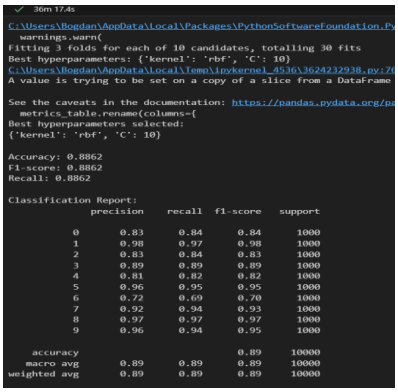
1. Timp executie: 2177.4 secunde

2. Rezultate pentru combinatii de hiperparametrii:

| param_C | param_kernel | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|
| 10 | rbf | **0.886** | 0.002 | 0.886 | **0.886** | **0.886** |
| 10 | poly | 0.883 | 0.002 | **0.883** | 0.883 | 0.883 |
| 20 | rbf | **0.886** | 0.002 | 0.885 | **0.886** | 0.885 |
| 20 | poly | 0.881 | 0.002 | 0.881 | 0.881 | 0.881 |
| 30 | rbf | 0.884 | 0.002 | 0.883 | 0.884 | 0.884 |
| 30 | poly | 0.88 | 0.003 | 0.88 | 0.88 | 0.88 |
| 40 | rbf | 0.883 | 0.002 | 0.882 | 0.883 | 0.882 |
| 40 | poly | 0.878 | **0.004** | 0.879 | 0.878 | 0.879 |
| 50 | rbf | 0.882 | 0.002 | 0.882 | 0.882 | 0.882 |
| 50 | poly | 0.878 | 0.003 | 0.878 | 0.878 | 0.878 |

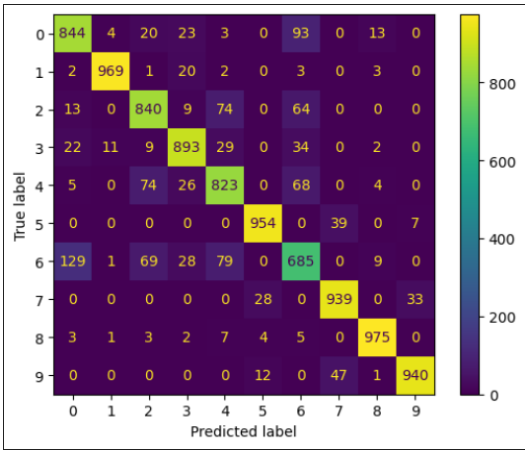
Combinatiile de hiperparametrii ce contin **rbf** obtin performante mai bune comparativ cu restul cum se poate observa din tabel.

3. Raport de clasificare si hiperparametrii optimi determinati:



```
✓ 36m 17.4s
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundation.Py
  warnings.warn(
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best hyperparameters: {'kernel': 'rbf', 'C': 10}
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_4536\3624232938.py:7
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pa
  metrics_table.rename(columns={
Best hyperparameters selected:
{'kernel': 'rbf', 'C': 10}

Accuracy: 0.8862
F1-score: 0.8862
Recall: 0.8862

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.84      0.84      1000
           1       0.98      0.97      0.98      1000
           2       0.83      0.84      0.83      1000
           3       0.89      0.89      0.89      1000
           4       0.81      0.82      0.82      1000
           5       0.96      0.95      0.95      1000
           6       0.72      0.69      0.70      1000
           7       0.92      0.94      0.93      1000
           8       0.97      0.97      0.97      1000
           9       0.96      0.94      0.95      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

4. Matrice de confuzie obtinuta:



Se poate observa o performanta mai buna a predictiilor, clasa **Sneaker**(7) avand acuratete mai mare comparative cu metodei de regresie logistica.

**c. Random Forest**

1. Timp executie: 445.6 secunde
2. Rezultate pentru combinatii de hiperparametrii:

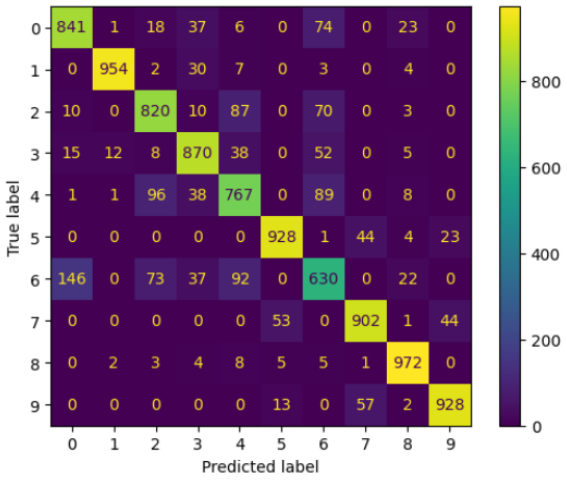
| param_n_estimators | param_max_depth | param_max_samples | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|---|
| 150 | 100 | 0.7 | **0.861** | 0.002 | **0.86** | **0.861** | **0.86** |
| 50 | 50 | 0.5 | 0.855 | 0.002 | 0.854 | 0.855 | 0.854 |
| 100 | 100 | 0.5 | 0.857 | **0.003** | 0.856 | 0.857 | 0.856 |
| 100 | 100 | 0.6 | 0.859 | 0.002 | 0.858 | 0.859 | 0.858 |
| 10 | 50 | 0.6 | 0.83 | 0.002 | 0.829 | 0.83 | 0.829 |
| 150 | 100 | 0.6 | 0.86 | 0.002 | 0.859 | 0.86 | 0.859 |
| 50 | 50 | 0.7 | 0.857 | 0.001 | 0.856 | 0.857 | 0.856 |
| 10 | 50 | 0.5 | 0.828 | 0.001 | 0.827 | 0.828 | 0.827 |
| 10 | 10 | 0.7 | 0.825 | 0.002 | 0.824 | 0.825 | 0.824 |
| 50 | 50 | 0.6 | 0.856 | 0.002 | 0.855 | 0.856 | 0.855 |
| 50 | 10 | 0.7 | 0.835 | 0.001 | 0.834 | 0.835 | 0.834 |
| 100 | 100 | 0.7 | 0.86 | 0.001 | 0.859 | 0.86 | 0.859 |
| 10 | 10 | 0.5 | 0.822 | **0.003** | 0.822 | 0.822 | 0.822 |
| 10 | 10 | 0.6 | 0.822 | 0.002 | 0.822 | 0.822 | 0.821 |
| 50 | 100 | 0.6 | 0.856 | 0.002 | 0.855 | 0.856 | 0.855 |
| 150 | 50 | 0.5 | 0.858 | 0.002 | 0.857 | 0.858 | 0.857 |
| 150 | 50 | 0.6 | 0.86 | 0.002 | 0.859 | 0.86 | 0.859 |
| 50 | 10 | 0.6 | 0.835 | 0.002 | 0.834 | 0.835 | 0.834 |
| 150 | 10 | 0.7 | 0.838 | 0.001 | 0.837 | 0.838 | 0.837 |
| 50 | 10 | 0.5 | 0.836 | 0 | 0.835 | 0.836 | 0.835 |

Se poate observa ca numarul de arbori utilizati pentru predictie afecteaza cel mai mult performanta si rezultatele obtinute.

3. Raport de clasificare si hiperparametrii optimi determinati:

```
✓ 7m 25.6s

Fitting 3 folds for each of 20 candidates, totalling 60 fits
Best hyperparameters: {'n_estimators': 150, 'max_samples': 0
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_4536\3624232938
A value is trying to be set on a copy of a slice from a Data

See the caveats in the documentation: https://pandas.pydata.
  metrics_table.rename(columns={
Best hyperparameters selected:
{'n_estimators': 150, 'max_samples': 0.7, 'max_depth': 100}

Accuracy: 0.8612
F1-score: 0.8612
Recall: 0.8612

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.84      0.84      1000
           1       0.98      0.95      0.97      1000
           2       0.80      0.82      0.81      1000
           3       0.85      0.87      0.86      1000
           4       0.76      0.77      0.77      1000
           5       0.93      0.93      0.93      1000
           6       0.68      0.63      0.65      1000
           7       0.90      0.90      0.90      1000
           8       0.93      0.97      0.95      1000
           9       0.93      0.93      0.93      1000

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000
```

4. Matrice de confuzie obtinuta:




Performanta obtinuta este mai slaba comparativ cu **SVM**, dar mai buna raportandu-ne la **Logistic Regression**.

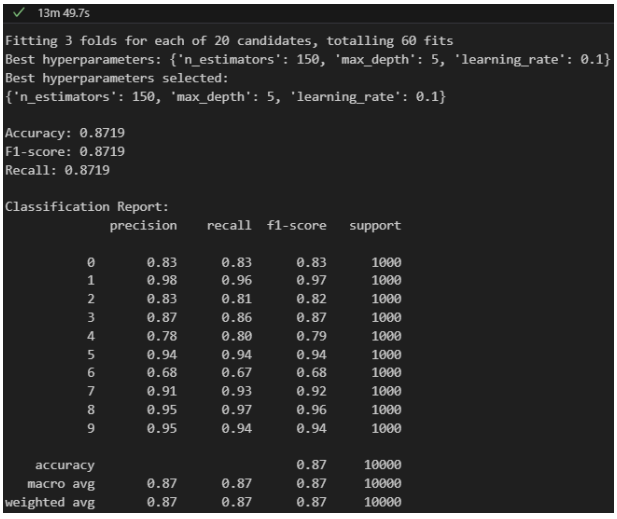
**d. Gradient Boosted Trees**

1. Timp executie: 829.7 secunde
2. Rezultate pentru combinatii de hiperparametrii:

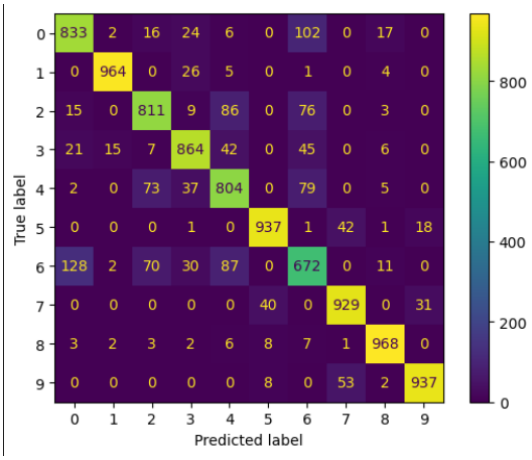
| param_n_estimators | param_max_depth | param_learning_rate | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|---|
| 150 | 7 | 0.01 | 0.847 | 0.003 | 0.845 | 0.847 | 0.845 |
| 100 | 5 | 0.05 | 0.856 | 0.002 | 0.855 | 0.856 | 0.855 |
| 50 | 3 | 0.05 | 0.811 | 0.004 | 0.809 | 0.811 | 0.809 |
| 50 | 5 | 0.1 | 0.855 | 0.002 | 0.854 | 0.855 | 0.854 |
| 50 | 3 | 0.01 | 0.764 | 0.001 | 0.764 | 0.764 | 0.762 |
| 150 | 3 | 0.05 | 0.843 | 0.002 | 0.842 | 0.843 | 0.842 |
| 100 | 7 | 0.05 | 0.866 | 0.002 | 0.865 | 0.866 | 0.866 |
| 150 | 7 | 0.05 | 0.872 | 0.002 | 0.872 | 0.872 | 0.872 |
| 50 | 5 | 0.05 | 0.84 | 0.003 | 0.839 | 0.84 | 0.839 |
| 50 | 7 | 0.1 | 0.866 | 0.003 | 0.865 | 0.866 | 0.865 |
| 100 | 3 | 0.01 | 0.784 | 0.002 | 0.782 | 0.784 | 0.782 |
| 100 | 5 | 0.01 | 0.823 | 0.004 | 0.821 | 0.823 | 0.821 |
| 150 | 5 | 0.01 | 0.83 | 0.003 | 0.828 | 0.83 | 0.828 |
| 150 | 3 | 0.01 | 0.796 | 0.003 | 0.794 | 0.796 | 0.794 |
| 50 | 7 | 0.05 | 0.854 | 0.002 | 0.853 | 0.854 | 0.853 |
| 100 | 5 | 0.1 | 0.869 | 0.002 | 0.868 | 0.869 | 0.868 |
| 50 | 5 | 0.01 | 0.812 | **0.005** | 0.81 | 0.812 | 0.81 |
| 100 | 7 | 0.1 | **0.876** | 0.002 | 0.875 | **0.876** | 0.875 |
| 150 | 5 | 0.1 | **0.876** | 0.002 | **0.876** | **0.876** | **0.876** |
| 50 | 3 | 0.1 | 0.831 | 0.004 | 0.829 | 0.831 | 0.83 |

Se observa ca numarul de arbori si adancimea acestora afecteaza cel mai mult metricile obtinute.

3. Raport de clasificare si hiperparametrii optimi determinati:



4. Matrice de confuzie obtinuta:

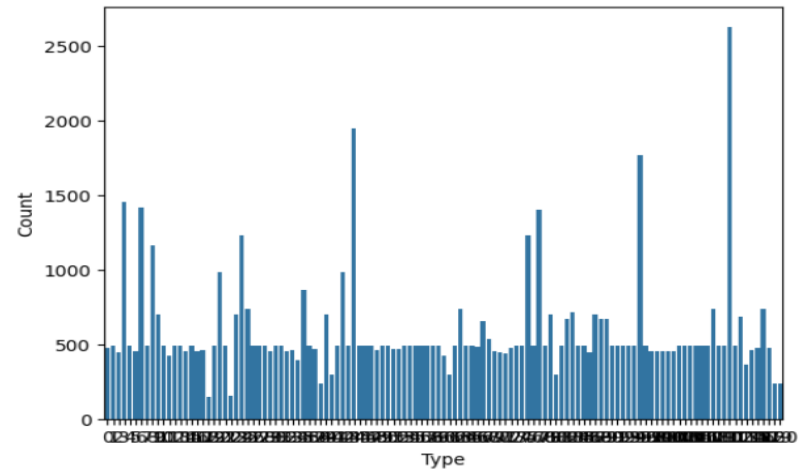


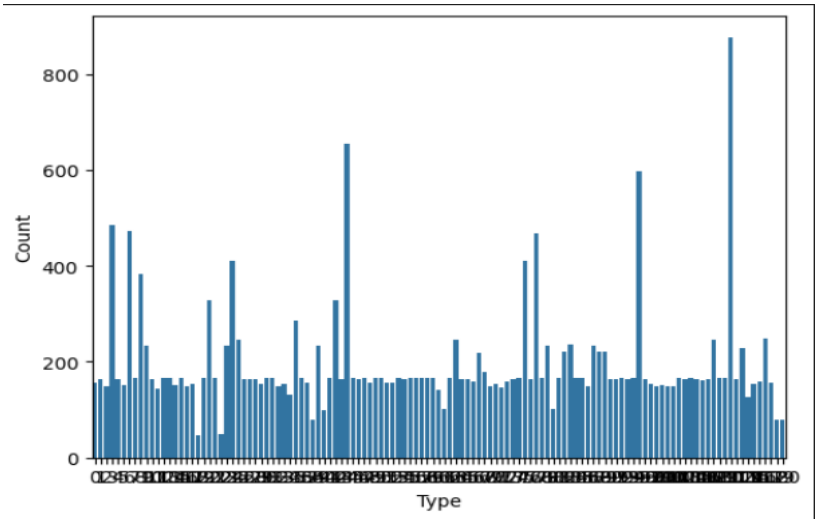

In afara de **SVM**, se pot observa cele mai bune performante pentru acest algoritm.

2. *Fruits-360*
   *EDA*

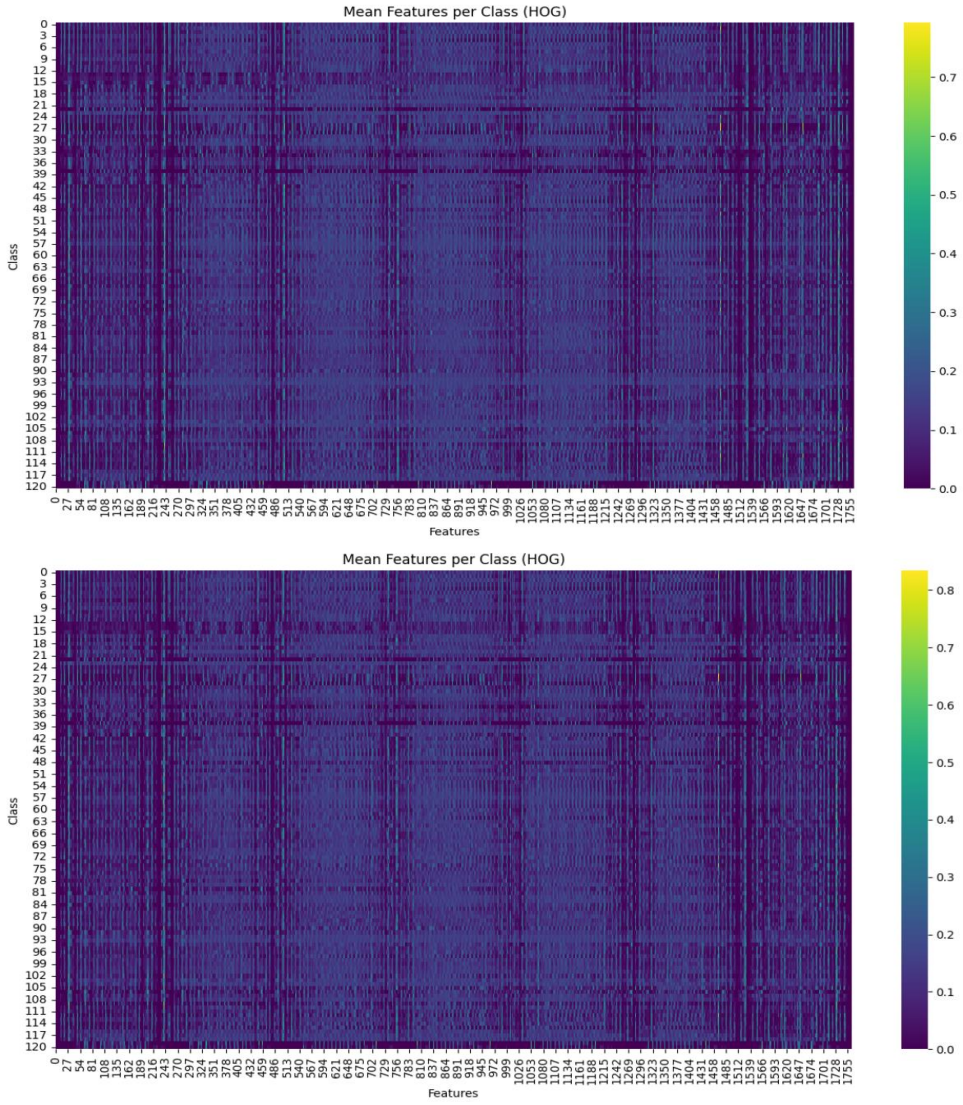
Metode alese pentru extragerea de attribute din imagini:

a. HOG(Histogram of Gradients)
   i. Alegere strict bazata in acest caz pe simplitatea metodei
   ii. Vizualizarile si statisticile cerute atat pe setul de antrenare cat si pe cel de testare:

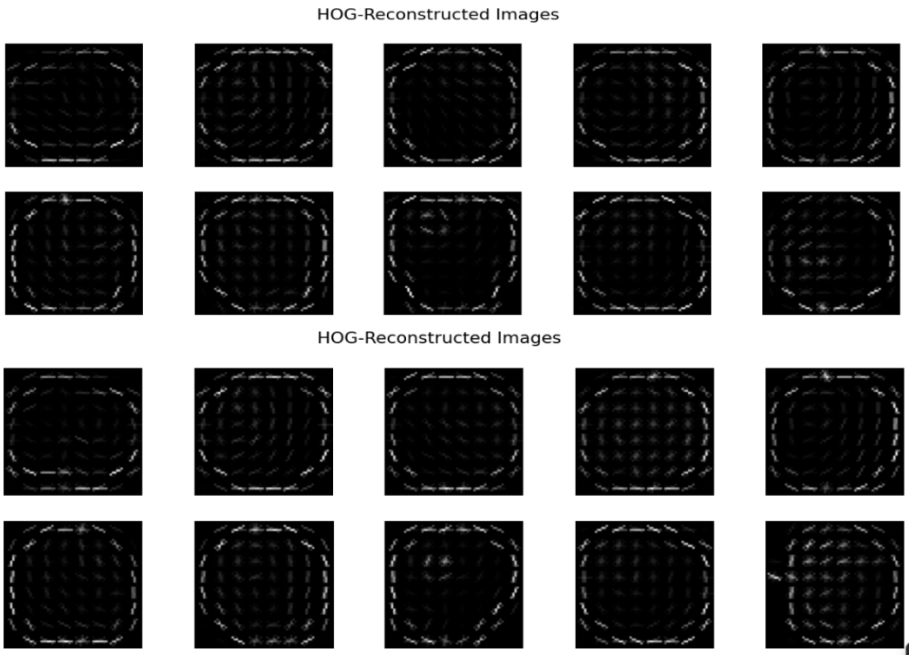


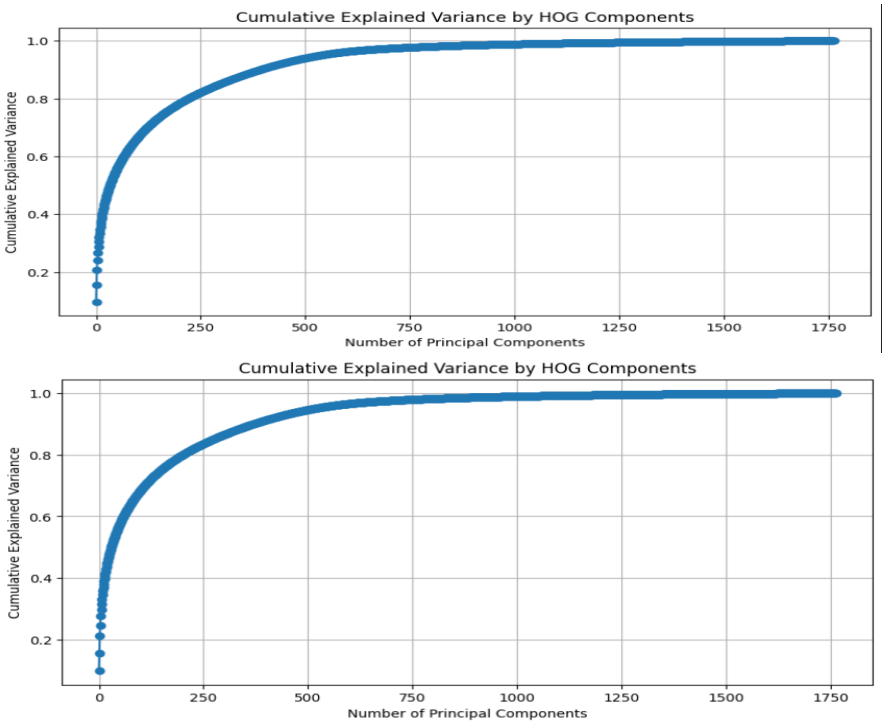

Observam un dezechilibru pentru anumite clase. In total am extras 121 de clase din cele 141 de directoare puse la dispozitie pentru setul de antrenare.




Atributele extrase pentru fiecare clasa in parte nu sunt tocmai cele mai relevante in urma utilizarii **HOG**, deoarece in cazul fructelor se tine cont si de culoarea acestora, dar **HOG** nu retine informatii legate de culoare.

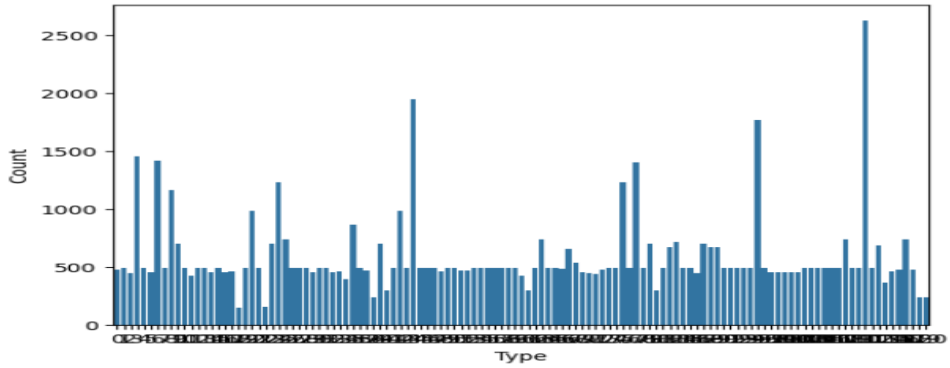


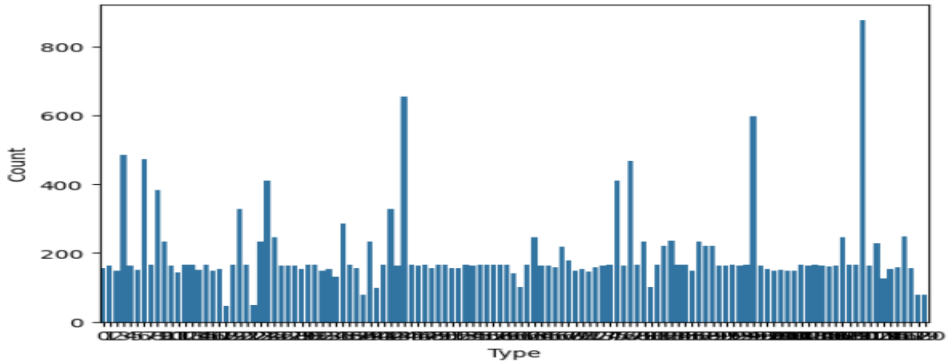

Imaginile reconstruite prezinta similaritati mari, datorita formei fructelor.

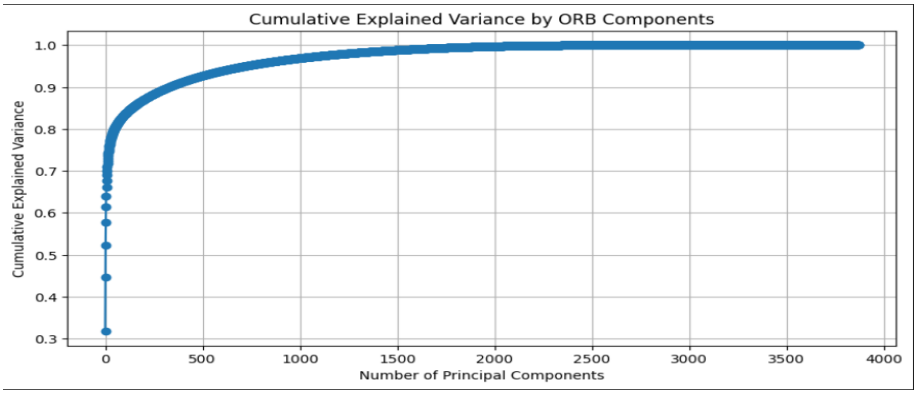


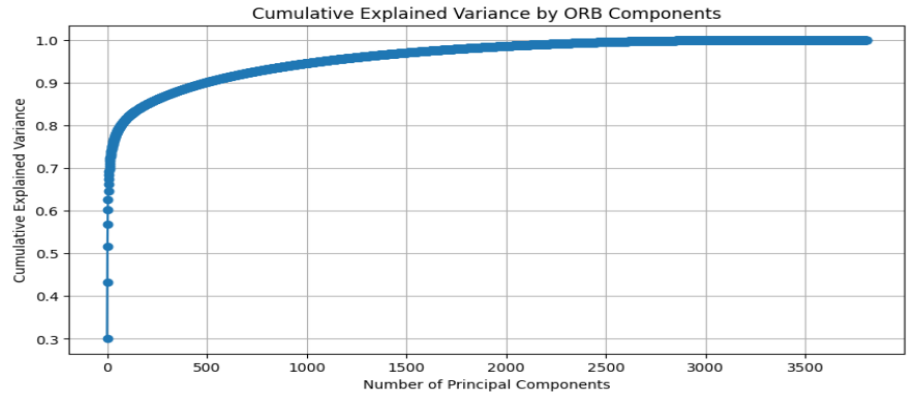

Se observa, exact ca la setul anterior de date, ca aceasta metoda necesita ~500 de atribute pentru varianta ridicata.

b. ORB(Oriented FAST and Rotated BRIEF)

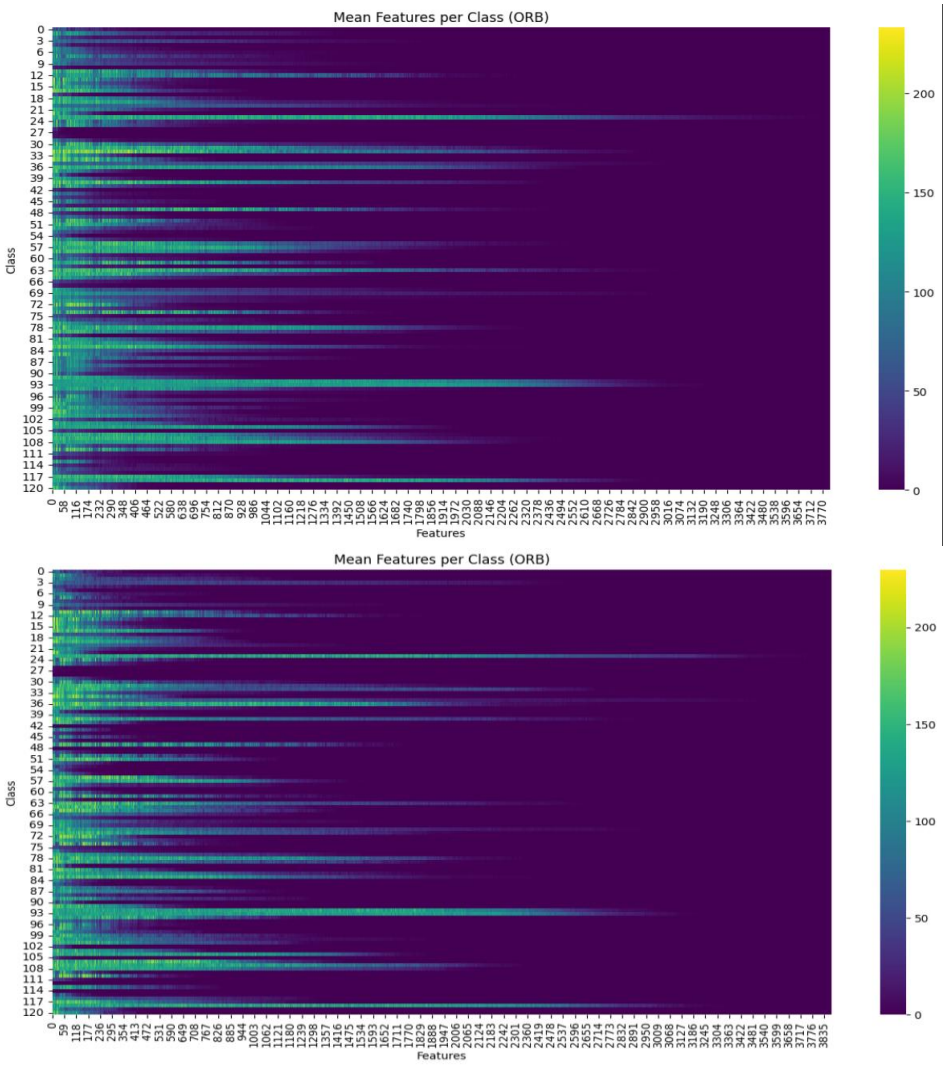


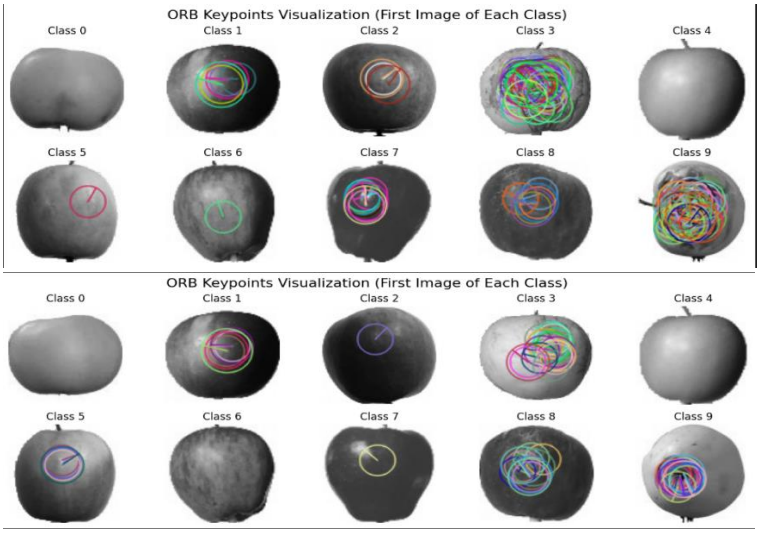




Comparativ cu **HOG**, aici putem observa o crestere mai rapida(dar pentru mai multe atribute).




Se poate observa din aceste 2 heatmap-uri ca majoritatea atributelor extrase cu **ORB** prezinta relevanta ridicata in raport cu imaginile(**ORB** tine cont atat de forma obiectului cat si de culoarea acestuia).




Imaginile reconstruite in urma utilizarii **ORB**.

## *Data Preprocessing*

Metode similare cu cele utilizate la setul de date anterior au fost folosite si aici.

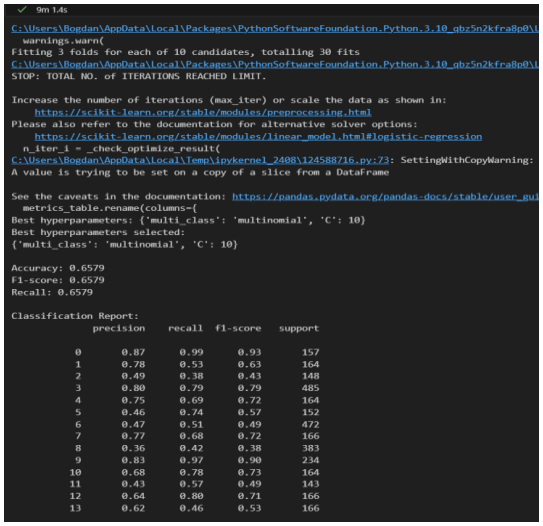
## *Models Training and Performance Counters*

a. **Logistic Regression**
   1. Timp executie: 541.4 secunde
   2. Rezultate pentru combinatii de hiperparametrii:

| param_C | param_multi_class | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|
| 10 | ovr | 0.394 | **0.055** | 0.442 | 0.422 | 0.415 |
| 10 | multinomial | **0.415** | 0.052 | **0.444** | **0.448** | **0.432** |
| 20 | ovr | 0.391 | 0.053 | 0.441 | 0.419 | 0.413 |
| 20 | multinomial | 0.414 | 0.052 | 0.442 | 0.447 | 0.431 |
| 30 | ovr | 0.39 | 0.054 | 0.442 | 0.418 | 0.413 |
| 30 | multinomial | 0.414 | 0.052 | 0.442 | 0.446 | 0.43 |
| 40 | ovr | 0.389 | 0.054 | 0.441 | 0.417 | 0.412 |
| 40 | multinomial | 0.413 | 0.051 | 0.441 | 0.445 | 0.429 |
| 50 | ovr | 0.39 | **0.055** | 0.442 | 0.418 | 0.412 |
| 50 | multinomial | 0.412 | 0.051 | 0.44 | 0.445 | 0.428 |

Se observa rezultate slabe in momentul utilizarii acestui algoritm din cauza utilizarii metodei **HOG** de extragere a atributelor(metoda ce nu tine cont si de culoarea fructelor).

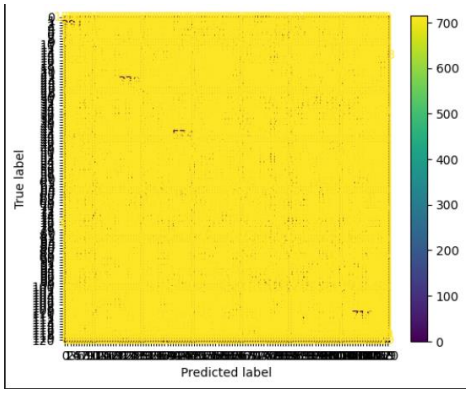
3. Raport de clasificare si hiperparametrii optimi determinati:



```
✓ 9m 1.4s
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\L
  warnings.warn(
Fitting 3 folds for each of 10 candidates, totalling 30 fits
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\L
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_2408\124588716.py:73: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gui
  metrics_table.rename(columns={
Best hyperparameters: {'multi_class': 'multinomial', 'C': 10}
Best hyperparameters selected:
{'multi_class': 'multinomial', 'C': 10}

Accuracy: 0.6579
F1-score: 0.6579
Recall: 0.6579

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       157
           1       0.78      0.53      0.63       164
           2       0.49      0.38      0.43       148
           3       0.80      0.79      0.79       485
           4       0.75      0.69      0.72       164
           5       0.46      0.74      0.57       152
           6       0.47      0.51      0.49       472
           7       0.77      0.68      0.72       166
           8       0.36      0.42      0.38       383
           9       0.83      0.97      0.90       234
          10       0.68      0.78      0.73       164
          11       0.43      0.57      0.49       143
          12       0.64      0.80      0.71       166
          13       0.62      0.46      0.53       166
```

Rezultatele generale de clasificare au fost mai bune fata de cele obtinute in tabelul prezentat anterior(probabil si pentru ca nu sunt visible toate clasele din raport).

4. Matrice de confuzie obtinuta:



**b. SVM**

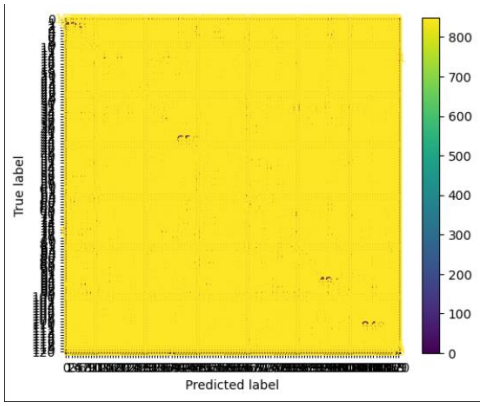
1. Timp executie: 3503.9 secunde
2. Rezultate pentru combinatii de hiperparametrii:

| param_C | param_kernel | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|
| 0.1 | rbf | 0.414 | **0.093** | 0.481 | 0.422 | 0.416 |
| 0.1 | poly | 0.314 | 0.065 | 0.444 | 0.312 | 0.324 |
| 1 | rbf | **0.521** | 0.06 | **0.582** | **0.557** | **0.543** |
| 1 | poly | 0.456 | 0.071 | 0.566 | 0.486 | 0.481 |

3. Raport de clasificare si hiperparametrii optimi determinati:

```
✓ 58m 23.9s
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundat
  warnings.warn(
Fitting 3 folds for each of 4 candidates, totalling 12 fits
Best hyperparameters: {'kernel': 'rbf', 'C': 1}
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_19272\124588716
A value is trying to be set on a copy of a slice from a Data

See the caveats in the documentation: https://pandas.pydata.
  metrics_table.rename(columns={
Best hyperparameters selected:
{'kernel': 'rbf', 'C': 1}

Accuracy: 0.8186
F1-score: 0.8186
Recall: 0.8186

Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       157
           1       0.84      0.68      0.75       164
           2       0.55      0.80      0.65       148
           3       0.81      0.88      0.85       485
           4       0.85      0.70      0.77       164
           5       0.65      0.78      0.71       152
           6       0.61      0.85      0.71       472
           7       0.97      0.98      0.98       166
           8       0.72      0.87      0.79       383
           9       0.90      1.00      0.95       234
          10       0.98      0.80      0.88       164
          11       0.69      0.72      0.71       143
          12       0.84      0.93      0.88       166
          13       0.65      0.67      0.66       166
          14       0.54      0.50      0.52       152
...
weighted avg       0.83      0.82      0.82     23619
```

4. Matrice de confuzie obtinuta:



### c. Random Forest

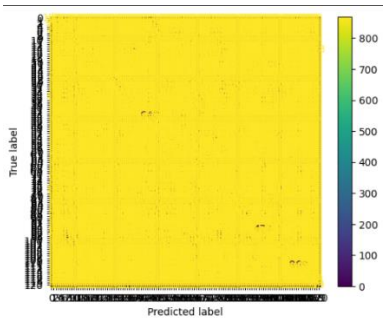
1. Timp executie: 1266.7 secunde
2. Rezultate pentru combinatii de hiperparametrii:

| param_n_estimators | param_max_depth | param_max_samples | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|---|
| 50 | 50 | 0.6 | 0.489 | **0.07** | 0.527 | 0.514 | 0.498 |
| 100 | 50 | 0.6 | 0.512 | 0.069 | 0.559 | 0.536 | 0.522 |
| 50 | 50 | 0.7 | 0.491 | **0.07** | 0.532 | 0.516 | 0.501 |
| 100 | 50 | 0.7 | 0.511 | 0.069 | 0.563 | 0.536 | 0.523 |
| 50 | 100 | 0.6 | 0.489 | **0.07** | 0.527 | 0.514 | 0.498 |
| 100 | 100 | 0.6 | **0.513** | 0.069 | 0.559 | **0.537** | 0.523 |
| 50 | 100 | 0.7 | 0.492 | **0.07** | 0.532 | 0.516 | 0.501 |
| 100 | 100 | 0.7 | 0.512 | 0.068 | **0.564** | **0.537** | **0.524** |

3. Raport de clasificare si hiperparametrii optimi determinati:

```
✓ 21m 6.7s
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n
  warnings.warn(
Fitting 3 folds for each of 8 candidates, totalling 24 fits
Best hyperparameters: {'n_estimators': 100, 'max_samples': 0.6, 'max_depth': 100}
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_2408\124588716.py:73: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  metrics_table.rename(columns={
Best hyperparameters selected:
{'n_estimators': 100, 'max_samples': 0.6, 'max_depth': 100}

Accuracy: 0.7732
F1-score: 0.7732
Recall: 0.7732

Classification Report:
              precision    recall  f1-score   support

           0       0.75      1.00      0.86       157
           1       0.80      0.65      0.72       164
           2       0.67      0.74      0.71       148
           3       0.71      0.87      0.79       485
           4       0.92      0.67      0.77       164
           5       0.60      0.66      0.63       152
           6       0.55      0.85      0.67       472
           7       0.95      0.97      0.96       166
           8       0.80      0.68      0.74       383
           9       0.86      1.00      0.93       234
          10       0.84      0.77      0.80       164
          11       0.67      0.66      0.66       143
          12       0.72      0.90      0.80       166
          13       0.69      0.63      0.66       166
          14       0.70      0.50      0.58       152
...
weighted avg       0.79      0.77      0.77     23619
```

4. Matrice de confuzie obtinuta:



### d. Gradient Boosted Trees

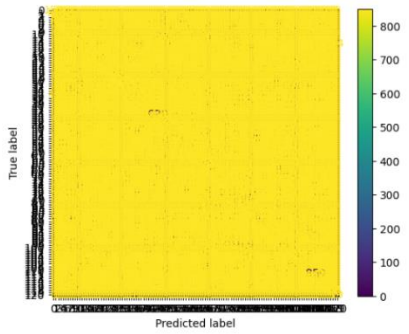
1. Timp executie: 3341.3
2. Rezultate pentru combinatii de hiperparametrii:

| param_n_estimators | param_max_depth | param_learning_rate | Accuracy (mean) | Accuracy (std) | Precision (mean) | Recall (mean) | F1-Score (mean) |
|---|---|---|---|---|---|---|---|
| 50 | 5 | 0.05 | 0.39 | 0.06 | 0.452 | 0.409 | 0.408 |
| 100 | 5 | 0.05 | 0.425 | **0.062** | 0.491 | 0.445 | 0.445 |
| 50 | 7 | 0.05 | 0.403 | 0.061 | 0.458 | 0.424 | 0.422 |
| 100 | 7 | 0.05 | 0.436 | 0.061 | 0.491 | 0.458 | 0.454 |
| 50 | 5 | 0.1 | 0.416 | 0.058 | 0.474 | 0.438 | 0.436 |
| 100 | 5 | 0.1 | **0.457** | 0.059 | **0.514** | 0.48 | **0.476** |
| 50 | 7 | 0.1 | 0.421 | 0.059 | 0.47 | 0.445 | 0.44 |
| 100 | 7 | 0.1 | **0.457** | 0.059 | 0.502 | **0.483** | 0.474 |

3. Raport de clasificare si hiperparametrii optimi determinati:

```
✓ 55m 41.3s
C:\Users\Bogdan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2
  warnings.warn(
Fitting 3 folds for each of 8 candidates, totalling 24 fits
Best hyperparameters: {'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1}
C:\Users\Bogdan\AppData\Local\Temp\ipykernel_2408\124588716.py:73: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  metrics_table.rename(columns={
Best hyperparameters selected:
{'n_estimators': 100, 'max_depth': 7, 'learning_rate': 0.1}

Accuracy: 0.7465
F1-score: 0.7465
Recall: 0.7465

Classification Report:
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       157
           1       0.83      0.65      0.73       164
           2       0.61      0.60      0.61       148
           3       0.79      0.81      0.80       485
           4       0.84      0.66      0.74       164
           5       0.64      0.78      0.70       152
           6       0.61      0.82      0.70       472
           7       0.87      0.90      0.88       166
           8       0.72      0.68      0.70       383
           9       0.85      1.00      0.92       234
          10       0.77      0.73      0.75       164
          11       0.54      0.55      0.55       143
          12       0.79      0.88      0.83       166
          13       0.69      0.62      0.65       166
          14       0.53      0.55      0.54       152
...
weighted avg       0.76      0.75      0.74     23619
```

4. Matrice de confuzie obtinuta:



## *Concluzii*

Rezultatele obtinute pe cele 2 seturi de date difera raportandu-ne la metricile obtinute, acestea depinzand de natura imaginilor si a distributiei lor. Cele mai bune rezultate au fost obtinute pe setul de date **Fashion-MNIST** datorita metodei de extragere utilizata(**HOG**) ce a determinat definirea formei hainelor prin care au fost extrase atribute relevante ce au ajutat la clasificare. Pe setul de date **Fruits-360** au fost obtinute rezultate mai putin precise datorita utilizarii aceleasi metode de extragere(am incercat sa utilizez **ORB** in continuare, dar era obtinuta o acuratete foarte scazuta, de ~15-20%). O alta caracteristica importanta in procesul de clasificare a fost timpul de executie pentru determinarea hiperparametrilor optimi. Cu cat numarul de combinatii de hiperparametrii era mai ridicat, cu atat timpul de executie era mai mare, situatie ce necesita multe resurse ale sistemului(era de preferat sa se ruleze pe un cluster tema sincer, nu aveam destul RAM in unele momente sa rulez). In acelasi timp pentru obtinerea unor performante mai mari ar fi trebuit selectate mai multe atribute in urma extragerii si a preprocesarii datelor. S-au ales 10% cele mai bune de fiecare data, utilizand metoda **SelectPercentile**(problema este ca daca numarul de atribute selectate era mai mare, timpul de executie era la randul lui mai mare, dar am observat pentru 20% atribute selectate o performanta mai buna a metricilor utilizate).

## *Observatii*

Tema a fost interesanta, ajutandu-ma sa capat cunostiinte noi(cum ar fi selectia de atribute din imagini si partea de cautare a hiperparametrilor optimi), cat si sa ma deprind mai bine cu Python-ul si module si biblioteci populare din acesta.