Analiza și Dezvoltarea unui Model pentru Clasificare Multi-Clasă - TIA 2024

Echipa: Vasile Ioan Teodor 331AC
Lungulescu Raul Adrian 331AC

Cuprins

1 Introducere

1.1 Contextul proiectului

Proiectul de fata isi propune sa implementeze o solutie in domeniul învățării automate și are ca scop dezvoltarea unui model de clasificare a imaginilor. Setul de date utilizat este format din imagini de articole vestimentare derivate din categoria Fashion MNIST, care reprezintă o variantă modificată a setului de date MNIST, folosit frecvent în cadrul cercetării în acest domeniu. Fashion MNIST conține imagini gri de 28x28 pixeli ale unor articole vestimentare, fiecare dintre acestea fiind asociată cu o etichetă ce indică tipul articolului de îmbrăcăminte sau încălțăminte.

În cadrul acestui proiect, imaginile din Fashion MNIST au fost preprocesate și transformate într-un set de caracteristici de dimensiune 784, obținute prin aplicarea unei proiecții aleatorii pe fiecare imagine. Fiecare vector de caracteristici reprezintă o versiune comprimata a imaginii originale și este folosit ca intrare pentru modelul de învățare automată. Acest proces de proiecție aleatorie este util pentru reducerea dimensiunii datelor, oferind în același timp suficiente informații relevante pentru problema de clasificare.

Proiectul are ca obiectiv dezvoltarea unui model de clasificare care să fie capabil să precizeze corect etichetele asociate imaginilor din setul de testare, pe baza vectorilor de caracteristici furnizați. Modelul antrenat poate fi evaluat pe un set de testare, iar predicțiile sale pot fi comparate cu etichetele corecte, măsurând astfel performanța modelului.

Importanța acestui proiect este data pe de o parte, de faptul ca acesta oferă o oportunitate de a aplica algoritmii utili in domeniul invatarii automate pentru procesarea imaginilor pentru o problemă reală, iar pe de altă parte, servește ca un exemplu pentru a înțelege cum se poate construi un model de clasificare a imaginilor, folosind algoritmi ce utilizeaza tehnici de reducere a dimensiunii și clasificare .

Proiectul se concentrează pe utilizarea algoritmilor de învățare automată pentru a extrage modele de date semnificative dintr-un set de date mare și complex, iar performanța modelului se va evalua prin măsuri de acuratețe, oferind astfel o oportunitate de a explora și compara diferite tehnici de învățare automată, inclusiv regresia logistică, SVM, KNN sau modelele bazate pe arbori de decizie. În plus, procesul de optimizare a modelului va include experimentarea cu diverse tehnici de regularizare și selecție a caracteristicilor pentru a preveni supraspecializarea și a îmbunătăți generalizarea modelului.

1.2 Obiectivele analizei

Scopul principal al acestei analize este de a construi și evalua un model de învățare automată pentru a prezice corect etichetele pentru fiecare exemplu din setul de date de testare. Obiectivele specifice ale proiectului includ:

- Încărcarea și preprocesarea seturilor de date de antrenament și testare.
- Implementarea unui model de învățare automată pentru clasificarea imaginilor pe baza

vectorilor de caracteristici.

- Antrenarea modelului utilizând setul de date de antrenament.
- Evaluarea performanței modelului pe setul de date de validare.
- Generarea predicțiilor pentru setul de date de testare și salvarea acestora într-un fișier CSV conform formatului cerut.
- Analiza performanței modelului și optimizarea acestuia pentru a îmbunătăți acuratețea predicțiilor.

2 Analiza setului de date

În cadrul acestui proiect, datele utilizate pentru antrenarea și testarea modelului provin dintr-un set de date bazat pe imagini de articole de îmbrăcăminte și încălțăminte, denumit *Fashion MNIST*. În această secțiune, vom analiza detaliat structura, caracteristicile și posibilele aspecte care pot influența analiza acestui set de date.

2.1 Descrierea setului de date

Setul de date utilizat în acest proiect denumit *Fashion MNIST*, conține imagini de 28x28 pixeli ale articolelor de îmbrăcăminte și încălțăminte. Setul Fashion MNIST conține 10 tipuri de articole de modă, iar fiecare imagine este etichetată corespunzător. Imaginile sunt redimensionate și transformate într-un vector de dimensiune 784 (28x28 pixeli), printr-o proiecție aleatorie.

Setul de date este împărțit în trei fișiere:

- **train.npz**: Acesta conține setul de antrenament și include două array-uri:
  - x_train: Un array 2D de dimensiune $(num\_samples, num\_features)$, în care fiecare rând reprezintă un vector de caracteristici pentru o imagine. Fiecare vector are 784 de elemente, corespunzând celor 28x28 pixeli ai imaginii originale.
  - y_train: Un array 1D de dimensiune $(num\_samples,)$, care conține etichetele (labelurile) corespunzătoare fiecărui vector de caracteristici. Fiecare etichetă este un număr întreg între 0 și 9, corespunzând unui tip de articol de modă.
- **test.npz**: Conține setul de testare, format doar din vectorii de caracteristici ale imaginilor, fără etichete. Formatul este similar cu x_train, dar fără etichetele asociate fiecărui test.
- **sample_submission.csv**: Un fișier CSV care servește drept ghid pentru realizarea predicțiilor. Acesta conține două coloane:
  - *Id*: Un identificator unic pentru fiecare test (corespunzător fiecărui rând din x_test).
  - *Label*: Eticheta prezisă de model pentru fiecare exemplu de test (aceasta va fi completată în urma predicțiilor modelului).

Tipuri de articole de modă: Etichetele sunt numerotate de la 0 la 9 și corespund diferitelor articole vestimentare sau de încălțăminte:

- 0: T-shirt/top
- 1: Trouser
- 2: Pullover
- 3: Dress
- 4: Coat
- 5: Sandal

- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

2.2 Structura si caracteristicile datelor

Setul de date este structurat astfel încât să permită o procesare eficientă în contextul unui algoritm de învățare automată. Mai jos sunt detaliate principalele componente ale datelor:

- **x_train**:
  - Este un array 2D de dimensiune $(num\_samples, num\_features)$, unde $num\_samples$ reprezintă numărul de imagini (exemple de antrenament), iar $num\_features$ reprezintă numărul de caracteristici (784 în acest caz, corespunzător numărului de pixeli din fiecare imagine de 28x28).
  - Fiecare rând al acestui array este un vector ce reprezintă o imagine din setul de date, unde fiecare element al vectorului reprezintă valoarea unui pixel după proiecția aleatorie.
- **y_train**:
  - Este un vector 1D de dimensiune $(num\_samples,)$, care conține etichetele pentru fiecare imagine. Aceste etichete sunt numere întregi între 0 și 9, care corespund tipurilor de articole vestimentare sau de încălțăminte (așa cum este detaliat în secțiunea anterioară).
  - Etichetele sunt folosite pentru a antrena modelul și pentru a-l învăța să asocieze fiecare vector de caracteristici cu o etichetă corectă.
- **x_test**:
  - Este un array 2D similar cu x_train, dar care conține doar caracteristicile imaginii, fără etichete. Aceste date sunt folosite pentru a face predicții.
- **sample_submission.csv**:
  - Conține două coloane: *Id* și *Label*. Coloana *Id* reprezintă identificatorul unic al fiecărei imagini de test din x_test, iar *Label* reprezintă eticheta prezisă de model.

2.2.1 Distribuția Claselor

Setul de date conține 10 clase, fiecare reprezentând un articol vestimentar. Graficul și lista de mai jos prezintă distribuția claselor:

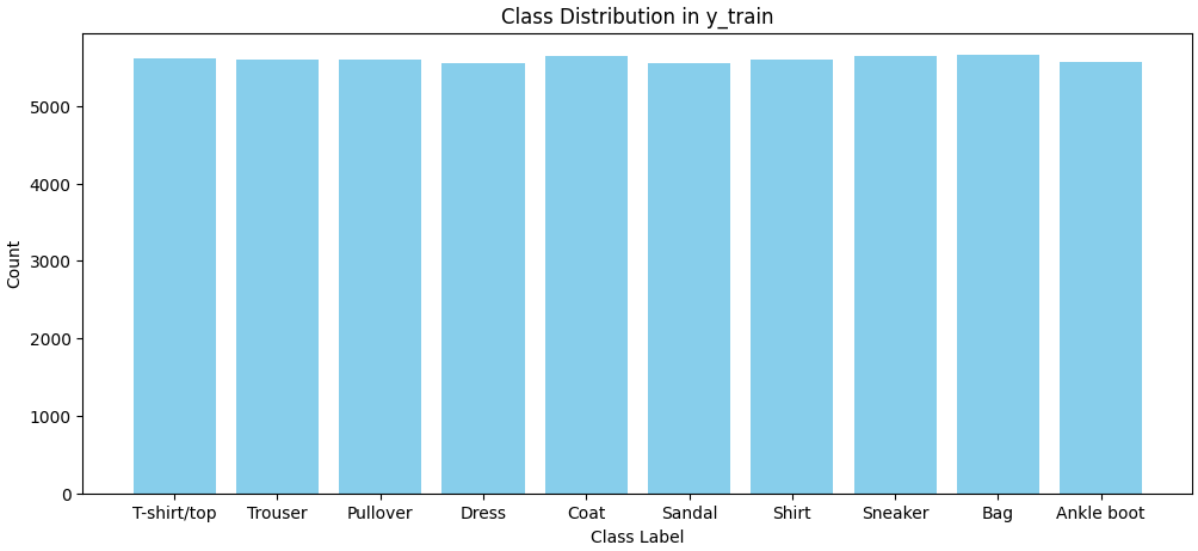

Figura 2.1: Distribuția claselor în setul de date.

Distribuția setului de date pe clase este următoarea:

- **T-shirt/top:** 5606 exemple
- **Trouser:** 5598 exemple
- **Pullover:** 5593 exemple
- **Dress:** 5551 exemple
- **Coat:** 5643 exemple
- **Sandal:** 5551 exemple
- **Shirt:** 5593 exemple
- **Sneaker:** 5641 exemple
- **Bag:** 5658 exemple
- **Ankle boot:** 5566 exemple

Setul de date este echilibrat și nu exista valori lipsă in setul de date.

2.3 Aspecte ce pot influenta analiza

Există mai mulți factori care pot influența analiza setului de date și performanța finală a modelului de învățare automată. Printre aceștia se numără:

- **Calitatea și preprocesarea datelor**:
  - Un factor important este calitatea datelor. Dacă setul de date conține imagini cu zdistorsiuni sau erori, acest lucru poate afecta performanța modelului. De asemenea, preprocesarea datelor ajuta pentru a transforma datele într-un format adecvat pentru învățarea automată. Preprocesarea poate include operații precum normalizarea, scalarea valorilor pixelilor sau aplicarea unor tehnici de augmentare a datelor.

- **Modele de învățare automată**:
  - Alegerea modelului de învățare automată este de asemnea foarte importanta. Modelele simple, cum ar fi regresia logistică sau SVM pot fi eficiente pentru clasificarea datelor. Experimentarea cu diverse tipuri de modele poate ajuta pentru a găsi cea mai bună abordare.
- **Echilibrul între clase**:
  - Distribuția etichetelor în setul de date este un alt factor ce poate influența performanța modelului. Dacă anumite clase sunt sub-reprezentate, modelul poate învăța să favorizeze predicțiile pentru clasele mai frecvente, iar acest lucru poate duce la un model cu acuratețe scăzută pentru clasele mai rare. În cazul nostru, clasele sunt echilibrat împărțite.
- **Hiperparametrii și regularizarea**:
  - Ajustarea hiperparametrilor modelului și aplicarea tehnicilor de regularizare poate ajuta pentru a preveni overfitting-ul și pentru a îmbunătăți generalizarea modelului la setul de date de testare.

Analiza setului de date este un proces esențial care trebuie să țină cont de multiple aspecte, de la calitatea datelor și alegerea modelului, până la tehnicile de preprocesare și regularizare.

3 Analiza statistică

3.1 Metode Statistice Descriptive

Pentru a înțelege mai bine structura și natura setului de date, au fost aplicate metode statistice descriptive. Această secțiune explică fiecare măsură utilizată, oferă interpretări relevante și include grafice ilustrative.

3.1.1 Măsuri ale Tendinței Centrale

Media (Mean)

Definiție: Media reprezintă valoarea medie a unui set de date, calculată ca suma tuturor valorilor împărțită la numărul total de observații. Reflectă „punctul central" al distribuției datelor.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Observații:

- Valorile medii sunt predominant în intervalul $[-5000, 5000]$, însă există câteva caracteristici care depășesc aceste limite.
- Aceste valori indică o variabilitate ridicată între caracteristici.
- Variabilitatea mare a mediilor sugerează necesitatea unei normalizări a datelor pentru a reduce influența caracteristicilor cu valori extreme.

Grafic:

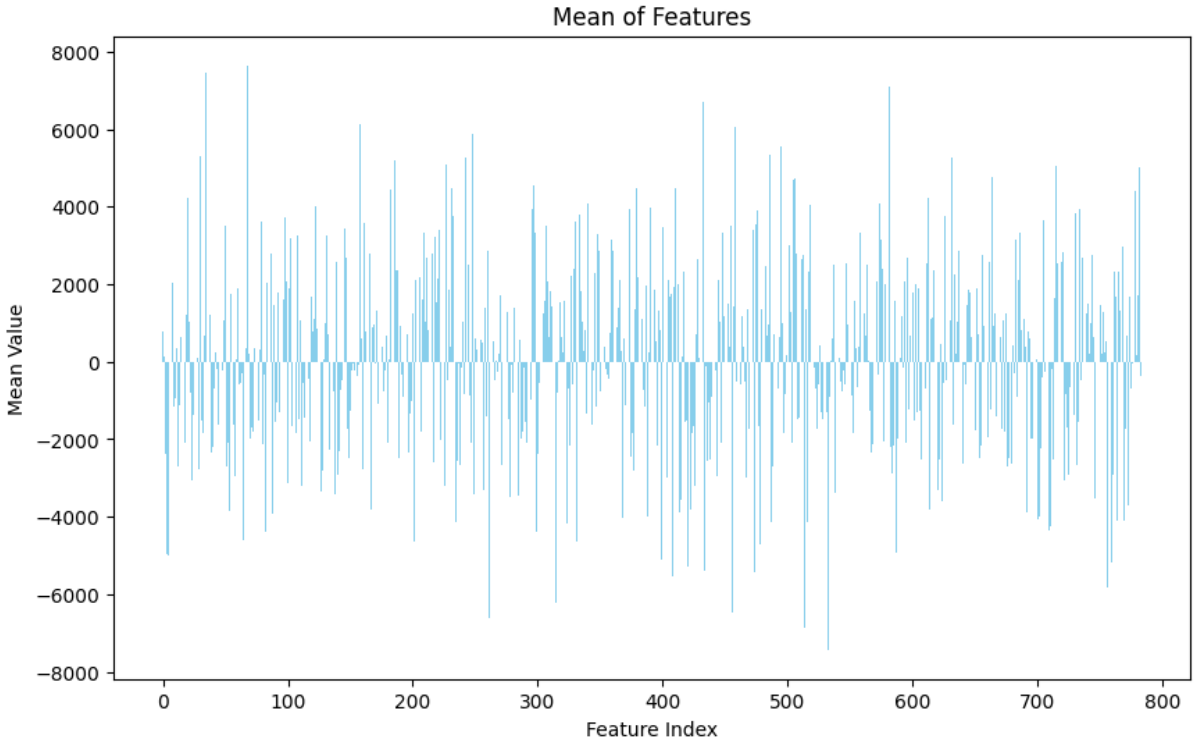

Figura 3.1: Distribuția valorilor medii pentru caracteristici.

3.1.2 Măsuri ale Dispersiei

Deviația Standard (Standard Deviation)

Definiție: Deviația standard măsoară dispersia valorilor în jurul mediei. O valoare mare indică o variabilitate ridicată, iar una mică sugerează valori apropiate de medie.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Observații:

- Valorile deviației standard sunt în general între 1500 și 2500, dar există și câteva caracteristici cu valori mult mai mari.
- Aceasta indică o răspândire semnificativă a datelor pentru anumite caracteristici.
- Dispersia ridicată a datelor justifică utilizarea metodelor de scalare/normalizare sau standardizare.

Grafic:

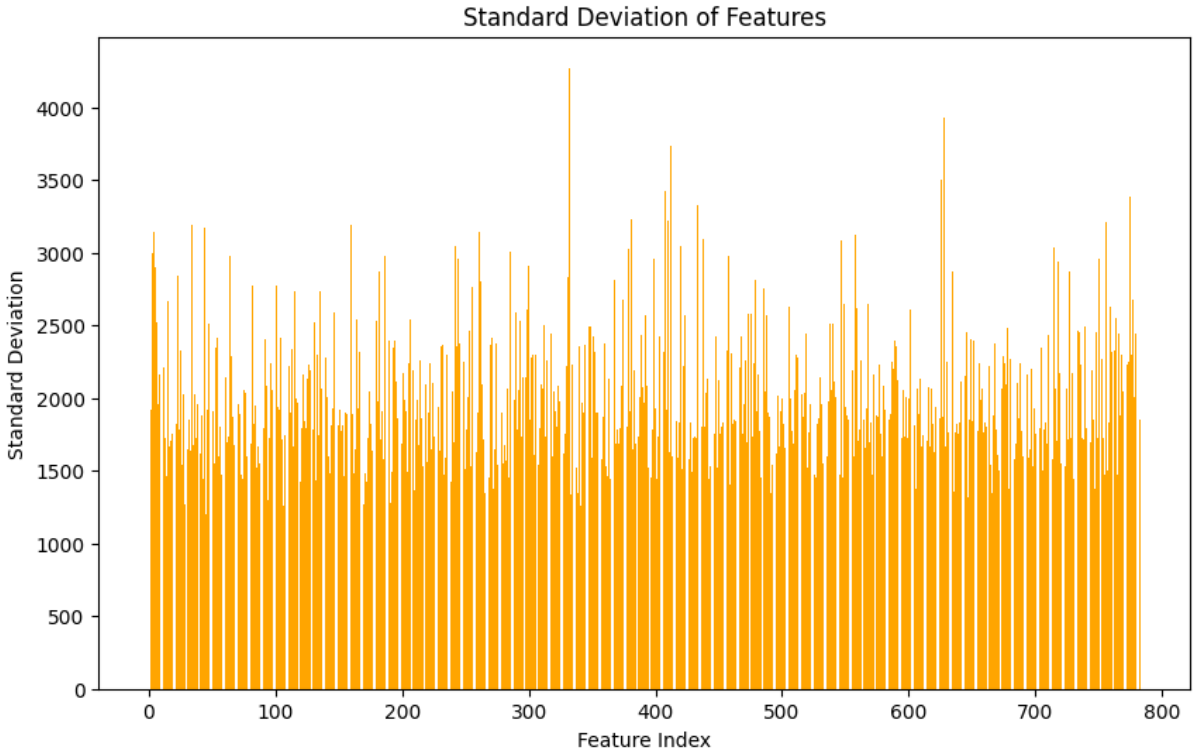

Figura 3.2: Distribuția deviației standard pentru caracteristici.

Range (Min, Max)

Definiție: Range-ul reprezintă diferența dintre valorile maxime și minime. Acesta oferă informații despre extinderea valorilor.

$$\text{Range} = \text{Max} - \text{Min}$$

Observații:

- Diferențele mari între valorile minime și maxime sugerează prezența unor outliers.

Grafic:

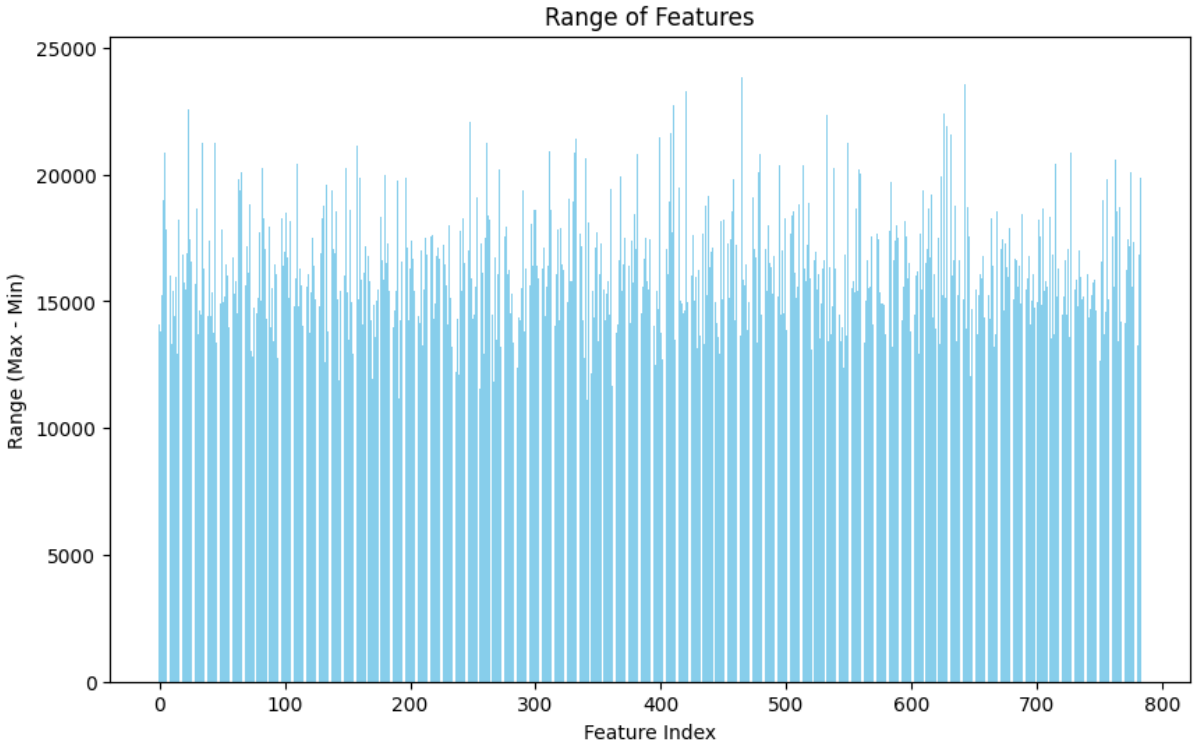

Figura 3.3: Distribuția valorilor minime și maxime.

3.1.3 Măsuri de Formă a Distribuției

Asimetria (Skewness)

Definiție: Asimetria măsoară cât de simetrică este distribuția datelor. O asimetrie pozitivă indică o coadă mai lungă spre dreapta, iar una negativă spre stânga.

$$\text{Skewness} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma} \right)^3$$

Observații:

- Majoritatea caracteristicilor au valori de asimetrie între $[-0.4, 0.4]$, ceea ce indică o distribuție aproape simetrică.
- Unele caracteristici prezintă asimetrii mai mari, sugerând posibile abateri.

Grafic:

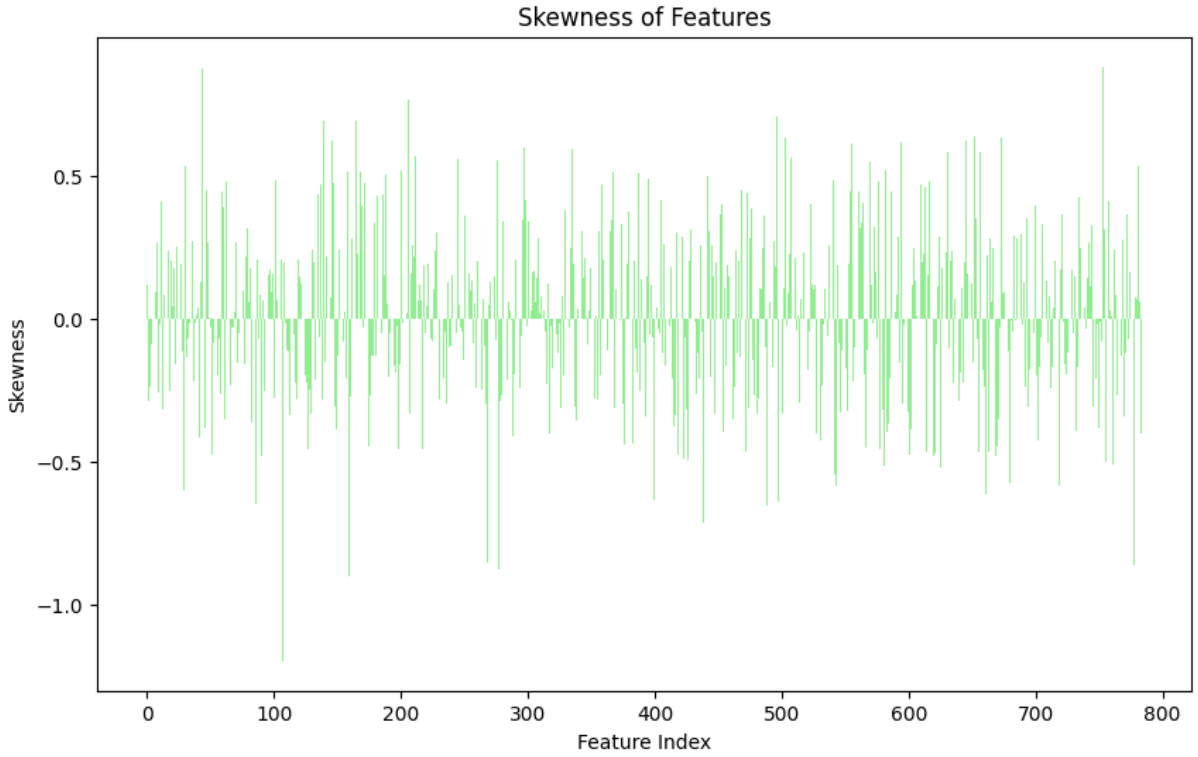

Figura 3.4: Distribuția asimetriei pentru caracteristici.

Curtosis (Kurtosis)

Definiție: Curtosis măsoară „grosimea" cozii distribuției. Valori mari sugerează prezența outlierilor.

$$\text{Kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma} \right)^4 - 3$$

Observații:

- Majoritatea caracteristicilor au valori de curtosis între $[-0.6, 0.6]$, dar câteva depășesc acest interval, indicând prezența valorilor extreme.

Grafic:

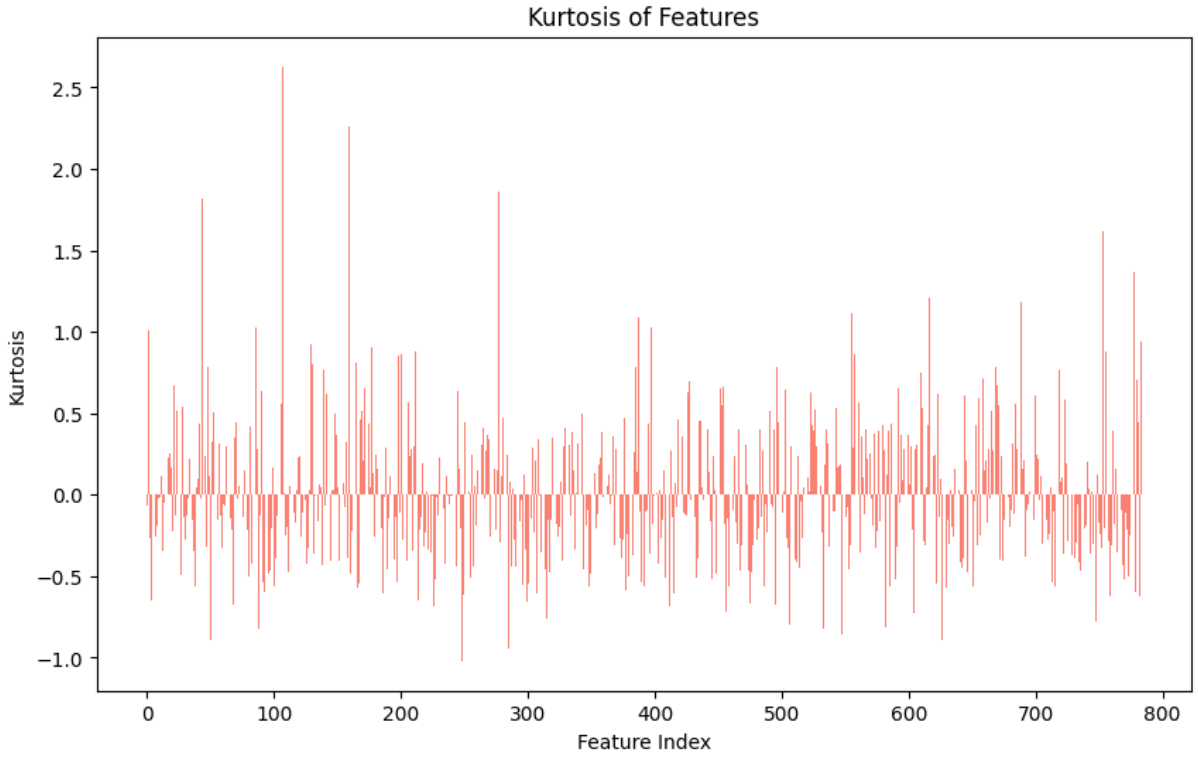

Figura 3.5: Distribuția curtosisului pentru caracteristici.

Așadar,

- Ar fi utilă o normalizare sau standardizare a datelor, având în vedere că valorile mediilor și deviațiilor standard variază semnificativ între caracteristici.
- Setul de date este echilibrat, ceea ce reduce riscul de dezechilibru al claselor în modelele de învățare automată.
- Majoritatea caracteristicilor au distribuții aproape simetrice, cu excepția unor caracteristici care prezintă asimetrie și curtosis mai mari, sugerând posibile valori extreme (outlieri).

3.2 Metode statistice avansate utilizate

3.2.1 Reducerea dimensionalitații folosind PCA

PCA (Principal Component Analysis) este o tehnică de reducere a dimensionalității care transformă datele originale într-un set de componente principale (PCs) ortogonale. Aceste componente reprezintă direcțiile cele mai semnificative ale variației din date și permit o reducere a numărului de variabile, păstrând informația esențială. Folosind PCA, reușim sa reducem dimensionalitatea setului de date pentru a le putea analiza sau vizualiza. De asemenea, poate ajuta la evitarea overfittingului.

3.2.2 PCA în 2D

Am aplicat PCA pentru a reduce datele la două componente principale, pentru a putea visualiza în plan distribuția datelor. Fiecare culoare reprezintă o clasă.

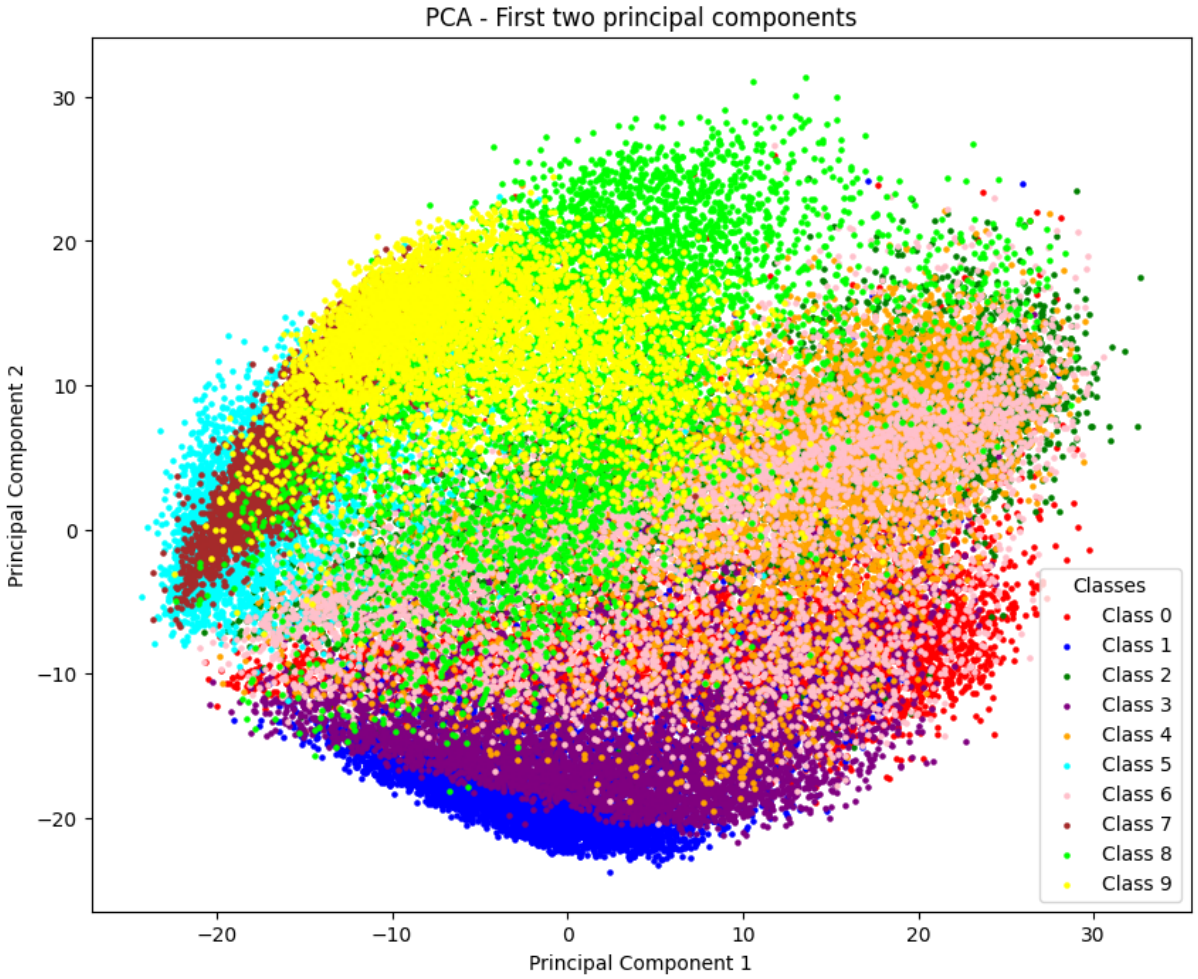

Figura 3.6: Vizualizarea PCA - Primele două componente principale (2D).

- Se observă o tendință de separare a datelor, însă sunt destule puncte care sunt împrăștiate
- PCA nu reușește să capteze complet structura internă a datelor cu doar 2 componente principale.

3.2.3 PCA în 3D

De asemenea, am afișat și in 3D datele

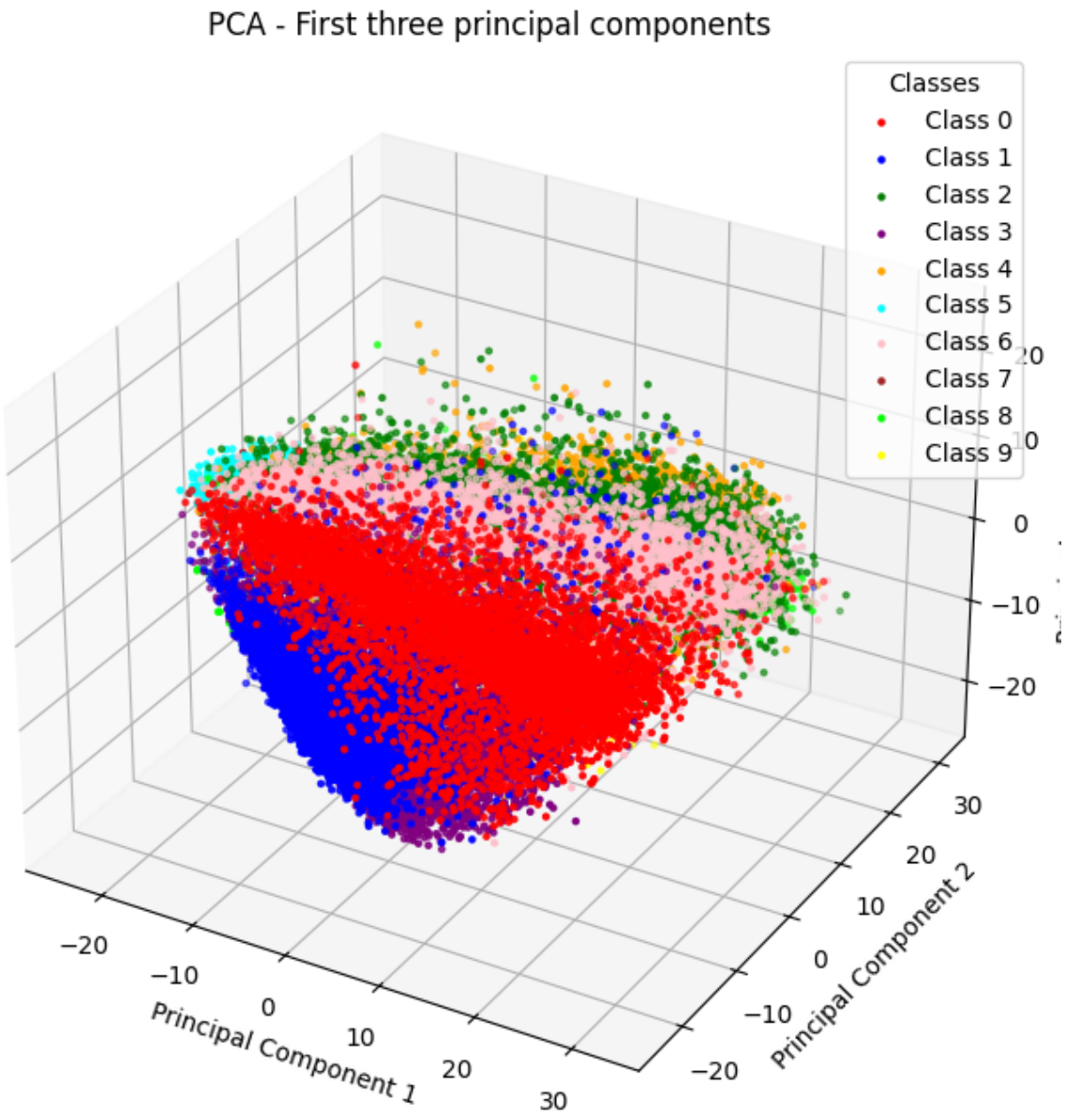

Figura 3.7: Vizualizarea PCA - Primele trei componente principale (3D).

Observații:

- Vizualizarea 3D permite o mai bună separare a datelor în comparație cu 2D.
- Devin mai clare diferențele între clase, dar încă se observă suprapuneri

4 Dezvoltarea modelului de invatare automata

Dezvoltarea unui model de învățare automată pentru clasificarea imaginilor pe baza setului de date derivat din Fashion MNIST implică mai mulți pași esențiali. Aceștia includ alegerea și implementarea unui algoritm de învățare automată, preprocesarea datelor, antrenarea modelului, evaluarea performanței și ajustarea hiperparametrilor pentru a îmbunătăți acuratețea acestuia.

4.1 Alegerea modelului

Pe baza analizei statistice, am decis sa alegem un model de învățare automată neliniar. Analiza a evidențiat că datele sunt distribuite relativ echilibrat între clase, dar cu unele distribuții neuniforme, iar variabilitatea mare în valorile caracteristicilor sugerează că un model liniar ar putea să nu fie suficient pentru a capta aceste relații complexe.

Astfel, am testat diferite modele, printre care K-Nearest Neighbors (KNN), Logistic Regression, Random Forest și Support Vector Machine (SVM), pentru a determina care este cel mai potrivit pentru clasificarea setului nostru de date.

După evaluarea performanței fiecărui model pe baza metodelor de **cross-validation** și a măsurătorilor de acuratețe, am ales *Support Vector Machine* (SVM One-vs-All) datorită capacității sale de a găsi o margine optimă între clase, chiar și în fața unui set de date cu variabilitate mare și distribuții non-liniare. De asemenea, SVM s-a comportat cel mai bine în fața distribuției echilibrate a claselor, oferind o performanță mai bună decât celelalte modele testate.

4.2 Metrici utilizate pentru evaluarea

In aceasta sectiune se vor prezenta principalele metode de evaluare a acuratetei algoritmilor prezentati mai sus.

- cross-validation - alinearea încrucișată împarte setul de date în mai multe seturi mai mici, numite fold-uri, și antrenează și testează modelul pe fiecare fold. Rezultatele sunt apoi date in medii pentru a oferi o estimare mai robustă a performanței modelului. In conetextul de fata s-a utilizat aceasta metoda pentru cei trei algortimi: logistic regression, knn, random forest in 5 fold-uri.
- Împărțirea în 80-20 se referă la o metodă comună de împărțire a datelor într-un set de antrenament și un set de testare. În mod specific, aceasta presupune utilizarea a 80 din date pentru antrenarea modelului și a 20 din date pentru evaluarea acestuia. Aceasta metoda s-a utilizat pentru SVM, pentru ca acest algoritm necesita un timp mare de executie. Totusi s-a realizat o comparatie si cu crros-validation ce va fi prezentata mai jos.

In urmatoarele se va prezenta rezultatul dat de cross-validation pe baza celor 5 fold-uri la cei trei algoritmi prezenti mai sus:

- **Logistic regression:** Cross-validation scores: [0.84901786 0.85205357 0.84678571 0.84758929 0.84794643] Mean cross-validation accuracy: **0.8487** Standard deviation of cross-validation accuracy: 0.0018
- **KNN** Cross-validation scores: [0.82982143 0.83794643 0.83160714 0.83223214 0.83767857] Mean cross-validation accuracy: **0.8339** Standard deviation of cross-validation accuracy: 0.0033
- **Random Forest** Cross-validation scores: [0.854375 0.85919643 0.85571429 0.85785714 0.85625] Mean cross-validation accuracy: **0.8567** Standard deviation of cross-validation accuracy: 0.0017

Cum s-a prezentat mai sus pentru comparatie se va prezenta rezultatele acuratetei SVM pe baza ambelor metode:

- **cu cross validation** Cross-validation scores: [0.8925 0.89732143 0.89232143 0.89571429 0.89544643] Mean cross-validation accuracy: **0.8947** Standard deviation of cross-validation accuracy: 0.0019
- **Cu impartirea 80-20** : Validation Accuracy: **0.8986**

Tabel cu comparatia intre cele doua metode:

Caracteristică	**Împărțirea 80-20**	**Cross-Validation**
Fiabilitate estimare	Mai puțin fiabilă (dependență de împărțirea aleatoare a datelor)	Mai fiabilă (mediată pe mai multe fold-uri)
Sensibilitate la împărțirea datelor	Mai sensibilă (setul de testare poate fi nereprezentativ)	Mai puțin sensibilă (folosește toate datele pentru testare)
Estimare corectă a performanței	Poate oferi o estimare optimistă sau pesimistă a acurateței	Oferă o estimare mai precisă a acurateței generale
Variația acurateței	Poate varia mult în funcție de setul de testare	Variație redusă datorită testării multiple
Performanță pe seturi dezechilibrate	Poate duce la estimări incorecte în cazul datelor dezechilibrate	Mai robustă pentru seturi dezechilibrate (asigură distribuție echilibrată în fold-uri)
Complexitate	Mai rapidă (un singur set de testare)	Mai complexă (testare de mai multe ori)

Tabela 4.1: Diferențe intre Acuratețea din Impărțirea 80-20 și Cross-Validation

4.3 Justificarea alegerii modelului

Aspect	**Avantaje**
Performanță	Eficient în spații cu dimensiuni mari și cu seturi de date mici.
Flexibilitate	Poate gestiona date neliniare utilizând funcții kernel.
Generalizare	Robust la overfitting, mai ales în spații cu dimensiuni mari.
Aplicații	Versatil în sarcini precum clasificarea textelor, recunoașterea imaginilor și bioinformatică.
Scalabilitate	Performanță fiabilă pe seturi de date mici spre medii.

Tabela 4.2: Avantajele SVM.

Aspect	Dezavantaje
Performanță	Consum intensiv de resurse computaționale pentru seturi de date mari.
Flexibilitate	Necesită ajustări atente ale parametrilor (de ex., tip kernel, regularizare).
Generalizare	Sensibil la zgomot și date etichetate incorect.
Aplicații	Lipsa ieșirilor probabilistice necesită metode suplimentare pentru scorurile de probabilitate.
Scalabilitate	Scalabilitate slabă pentru seturi de date foarte mari din cauza constrângerilor de timp și me

Tabela 4.3: Dezavantajele SVM.

SVM fiind folosit pentru modele liniare se utilizeaza kernel pentru ca setul de date este neliniar, acesta poate transforma datele non-liniare într-un spațiu înalt pentru a le face separabile. Funcția de pierdere SVM este adesea combinată cu un termen de regularizare pentru a preveni overfitting-ul. Tot odata, s-a utilizat diversi parametrii pentru a ajusta acuratetea modelului astfel:

- Parametrul C este un parametru de regularizare care controlează trade-off-ul între maximizarea marginii (separarea între clase) și minimizarea erorilor de clasificare. Rolul lui C este de a regla compromisurile dintre o marjă mai largă (generalizare) și penalizarea greșelilor de clasificare (precizia pe datele de antrenament).
- Un C mai mare presupune ca modelul va încerca să minimizeze cât mai mult erorile de clasificare pe datele de antrenament. Asta înseamnă că modelul va învăța să fie foarte precis pe datele de antrenament, iar acuratețea acestuia va fi mai mare. Totusi un C mai mare poate duce la overfitting.
- Gamma influențează forma kernelului utilizat pentru proiecția datelor într-un spațiu de dimensiuni mai mari. Acesta controlează influența unui punct de antrenament asupra altora în procesul de separare a claselor.
- Dacă gamma este 0.01, modelul va fi mai simplu și va putea generaliza mai bine pe seturile de testare, dar s-ar putea să nu învețe detaliile fine din setul de antrenament.

Parametrii alesi sunt svm classifier = SVC(kernel='rbf', C=10.0, gamma=0.01)

Acuratetea obtinuta la acest model se apropie de 90

Pentru a obtine un model cat mai eficient s-a realizat o comparatie cu diferiti algoritmi cu parametrii specifici pentru o comparatie cat mai buna:

- **KNN classifier**:
  - knn classifier = KNeighborsClassifier(neighbors=5, weights='distance'), unde **neighbors** reprezinta numarul de vecini pe care un punct ii foloseste pentru a determina clasa sa, se alege acest numar 5 pentru ca setul este format din 28x28 date intr-un vector; **weights='distance'** scade importanta vecinilor mai indepartati, ducand la cresterea acuratetei datorita pozitionarii neliniare a datelor.

  Acuratetea obtinuta la acest model se apropie de 0.8339, timpul de executie al algoritmului este unul scazut.
- **LogisticRegression**:
  - log reg classifier = LogisticRegression(solver='lbfgs', multiclass='multinomial', maxiter=1000), unde **solver='lbfgs'** Limited-memory Broyden–Fletcher–Goldfarb–Shanno este util in seturi mari de date si in multiclasing; **multiclass='multinomial'** utili-

zează probabilitatea multinomială pentru a rezolva problema în întregime; **maxiter=1000** este numarul de iteratii si pentru seturi mari de date se alege 1000 pentru ca algoritmul sa ajunga la convergenta.

Acuratetea obtinuta la acest model se apropie de 0.8484. Timpul de executie este mic.

- **Random Forest classifier**
  - rf classifier = RandomForestClassifier(estimators=100, maxdepth=None, randomstate=42, njobs=-1)), unde **estimators=100** este nr. de arbori de decizie, unul mai mare aduce o acuratete mai mare. **maxdepth=None** limitează adâncimea maximă a fiecărui arbore. Restul parametrilor nu influenteaza direct acuratetea

Acuratetea obtinuta la acest model se apropie de 0.8567. Timpul de executie este mai ridicat, dar ofera o acuratete mai ridicata.

5 Concluzii si recomandari

5.1 Rezumatul procesului

Pe parcursul proiectului curent, s-a realizat o serie de pași pentru dezvoltarea unui model de învățare automată destinat clasificării imaginilor pe baza setului de date obtinut din Fashion MNIST. S-a început cu analiza statistică a datelor, care a subliniat necesitatea normalizării și standardizării, precum și echilibrul între clase. Această analiză preliminară a ilustrat distribuțiile caracteristicilor și posibilele valori extreme, sugerând utilizarea unor metode robuste pentru preprocesarea datelor din set.

S-a aplicat ca metoda de reducere a dimensionalității PCA pentru vizualizarea și înțelegerea structurii interne a datelor. PCA a dat perspective utile asupra conexiunilor dintre caracteristici, permițând evidențierea principalelor direcții ale variației din setul de date. În special, vizualizările 2D și 3D au ajutat la identificarea unor modele de separare între clase, cu toate ca unele ofera suprapuneri persistente.

În etapa de dezvoltare a modelului, s-au evaluat mai mulți algoritmi de învățare automată, printre care K-Nearest Neighbors (KNN), Logistic Regression, Random Forest și Support Vector Machine (SVM). Pentru fiecare dintre aceștia, s-au optimizat hiperparametrii și s-a efectuat evaluări pe baza unor metode clasice de validare încrucișată. Pe baza rezultatelor obținute, am selectat SVM ca fiind modelul cel mai performant pe baza acuratetei.

5.2 Discutie asupra rezultatelor

Modelul SVM, configurat cu un kernel RBF, parametrul $C = 10.0$, și gamma $= 0.01$, a dat o acuratețe de aproximativ 90%. Această performanță mai ridicata a demonstrat capacitatea algoritmului de a captura relațiile complexe dintre caracteristici și de a generaliza bine pe date noi. În comparație, alte modele au oferit următoarele performanțe:

- **KNN**: Acuratețe de 83.39%, cu un timp de execuție scăzut, dar sensibil la distribuția datelor și la zgomot.
- **Logistic Regression**: Acuratețe de 84.84%, demonstrând o bună generalizare pentru datele liniare, dar limitări în fața structurilor neliniare.
- **Random Forest**: Acuratețe de 85.67%, cu o execuție mai lentă, dar beneficii în analiza datelor complexe și robustețe în fața zgomotului.

Vizualizările PCA au arătat o tendință generală de separare între clase, dar și limitările reducerii dimensionalității pentru captarea întregii structuri a datelor. PCA 3D a îmbunătățit claritatea relațiilor dintre clase, dar a evidențiat nevoia unor modele avansate pentru a trata complexitatea datelor.

5.3 Propuneri pentru imbunatatiri viitoare

Pentru a îmbunătăți performanța modelului și procesul general, se recomanda următoarele:

- **Explorarea metodelor de augmentare a datelor**: Crearea unor exemple sintetice prin tehnici precum rotație, scalare sau zgomot artificial poate îmbunătăți diversitatea datelor și performanța modelului.
- **Optimizarea automată a hiperparametrilor**: Implementarea unor tehnici precum Grid Search sau Bayesian Optimization ar putea rafina selecția parametrilor pentru o performanță îmbunătățită, dar aceasta metoda necesita un timp mare de procesare in executie si resurse hardware ridicate.
- **Explorarea tehnicilor alternative de reducere a dimensionalității**: Algoritmi precum t-SNE sau UMAP, superiori PCA ar putea oferi o înțelegere mai detaliată a distribuției.
- **Testarea pe seturi de date adiționale**: Aplicarea modelului pe alte seturi de date similare ar evalua capacitatea sa de generalizare și robustețe.
- **Evaluarea impactului zgomotului**: Investigarea performanței modelului în prezența datelor zgomotoase și optimizarea în consecință.

Prin implementarea acestor propuneri, proiectul ar putea atinge niveluri mai înalte de performanță și relevanță practică în aplicațiile de clasificare a imaginilor, crescand acuratetea modelului.