

# German Credit Data -MLP report-

## 1. Introducere

Scopul proiectului este de a antrena un sistem inteligent care să poată prezice dacă un client este unul „bun” sau „rău” din punct de vedere financiar, folosind o bază de date publică referitoare la creditele acordate de o bancă din Germania. Proiectul abordează o problemă de clasificare binară, folosind algoritmul Multilayer Perceptron (MLP), cu scopul de a evalua performanța pentru diferite arhitecturi și setări de hiperparametri.

---

## 2. Descrierea Bazei de Date

Baza de date utilizată este *german.data-numeric*, un fișier cu versiunea numerică a dataset-ului German Credit, care conține 1000 de eșantioane și 24 de caracteristici numerice. Fiecare instanță reprezintă un client, iar ultima coloană indică dacă acesta este considerat un client bun (1) sau rău (2) din punct de vedere al rambursării creditului.

Caracteristicile au fost preprocesate numeric și acoperă informații despre:

- Statutul contului curent
- Istoricul de credit
- Suma cerută
- Scopul creditului
- Venituri, loc de muncă, stare civilă etc.

Setul de date nu conține valori lipsă, ceea ce permite aplicarea directă a tehnicilor de scalare și echilibrare a claselor.

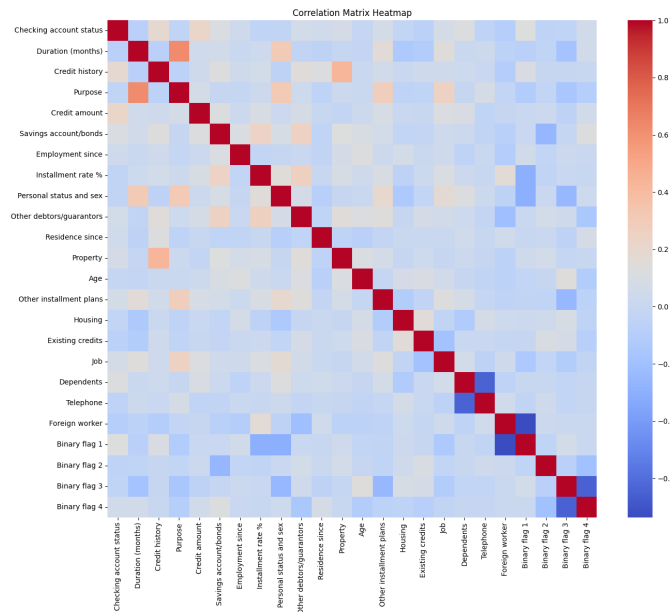
## 3. Procesarea datelor

În urma analizei distribuției variabilei țintă, am observat un dezechilibru semnificativ: 700 de exemple au fost etichetate ca „bun” (1) și doar 300 ca „rău” (2). Acest dezechilibru a influențat negativ performanța inițială a modelului, în special în ceea ce privește identificarea corectă a clasei minoritare. Pentru a contracara acest efect, am aplicat tehnica **SMOTE**, care generează exemple sintetice pentru clasa defavorizată, echilibrând astfel setul de date.

Inițial am folosit împărțirea clasică train-test-split, dar aceasta a dus la overfitting: modelul performa bine pe datele de antrenare, dar slab pe cele de testare. Pentru a obține o estimare mai robustă a performanței, am înlocuit această metodă cu validare încrucișată stratificată (**StratifiedKFold**, cu 5 folduri), asigurând o distribuție proporțională a claselor în fiecare subset.

De asemenea, am aplicat **standardizarea** caracteristicilor cu **StandardScaler**, deoarece rețelele neuronale sunt sensibile la variații mari între scările atributelor.

Pentru a analiza posibilele redundanțe dintre caracteristici, am generat o **matrice de corelație**. Majoritatea corelațiilor dintre atribute sunt slabe (valori apropiate de 0), ceea ce sugerează o relativă independență între ele. Nu s-au identificat perechi de atribute cu corelație ridicată ( $|r| > 0.95$ ), așa că am decis să păstrez toate caracteristicile.



## 4. Evaluarea modelului

După aplicarea SMOTE și a validării încrucișate, performanța modelului s-a îmbunătățit considerabil, în special pentru clasa minoritară. Pentru o imagine clară asupra predicțiilor, am generat o **matrice de confuzie**, care indică:

- 566 exemple corect clasificate ca „bune” (clasa 1),
- 239 exemple corect clasificate ca „rele” (clasa 2),
- 134 exemple din clasa 1 greșit clasificate ca 2,
- 61 exemple din clasa 2 greșit clasificate ca 1.

Pentru o evaluare mai exactă a performanței, am calculat următorii indicatori:

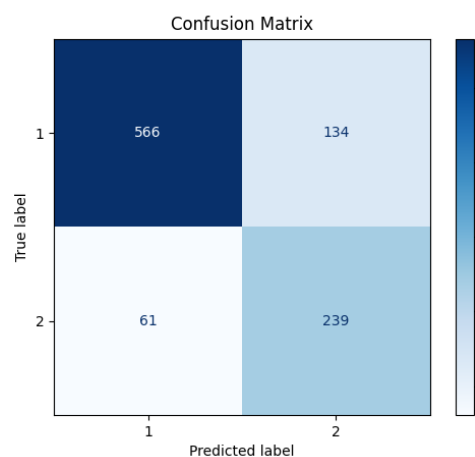
- **Acuratețe:** 0.81
- **Precizie:** 0.90 (clasa 1), 0.64 (clasa 2)
- **Recall:** 0.81 (clasa 1), 0.80 (clasa 2)
- **F1-score:** 0.85 (clasa 1), 0.71 (clasa 2)
- **Media macro:** Precizie: 0.77, Recall: 0.80, F1-score: 0.78
- **Media ponderată:** Precizie: 0.82, Recall: 0.81, F1-score: 0.81
- **Costul total al clasificărilor greșite:** 439

Pentru a înțelege care atribute au contribuit cel mai mult la deciziile modelului, am analizat **importanța caracteristicilor** folosind Permutation Importance (cu SMOTE și validare încrucișată). Cele mai influente variabile au fost:

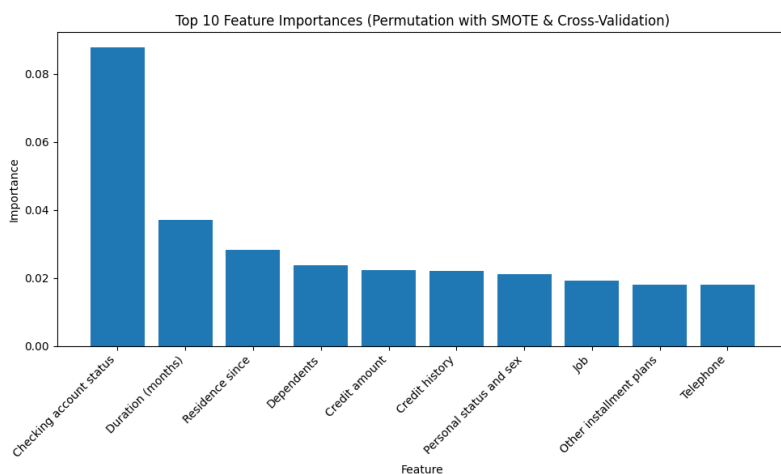
- **Checking account status**, care s-a detașat clar — fiind relevantă logic, deoarece reflectă stabilitatea financiară;

- urmată de **Duration (months)** și **Residence since**, dar și **Dependents**, **Credit amount** și **Credit history**.

Deși unele atribute au avut o importanță scăzută, niciuna nu a avut valoare zero, motiv pentru care am păstrat toate caracteristicile în modelul final.



**Matrice de confuzie**



**Importanța atributelor**

## 5. Precizari finale

Pentru antrenarea modelului, am testat mai multe configurații de rețele neuronale MLP, variind numărul de straturi ascunse, numărul de neuroni pe strat și rata de învățare. Procesul de selecție a hiperparametrilor a fost realizat folosind un script dedicat (**combinatii\_posibile.py**), care a aplicat validare încrucișată (cross-validation) cu 5 folduri și a evaluat fiecare combinație folosind F1-macro. Cele mai bune rezultate au fost obținute cu o arhitectură de tip (24, 24) și un learning rate de 0.01, această configurație atingând o acuratețe medie de aproximativ 70%.

Pentru a obține o imagine cât mai detaliată a performanței finale, modelul cu cei mai buni hiperparametri a fost antrenat și evaluat din nou în scriptul principal (**mlp\_model.py**). Deși acest script folosește din nou validare încrucișată, scopul său nu este selecția modelului, ci doar afișarea performanței obținute cu configurația optimă, pentru raportare și analiză. Dintre cele 5 modele antrenate prin cross-validation, a fost selectat cel cu cea mai bună acuratețe pe setul său de validare, pentru a estima performanța maximă posibilă a modelului final cu hiperparametrii aleși anterior. Această alegere reflectă intenția de a evidenția potențialul maxim al modelului în condiții ideale, chiar dacă rezultatele pot fi ușor supraestimate față de media generală a performanței.