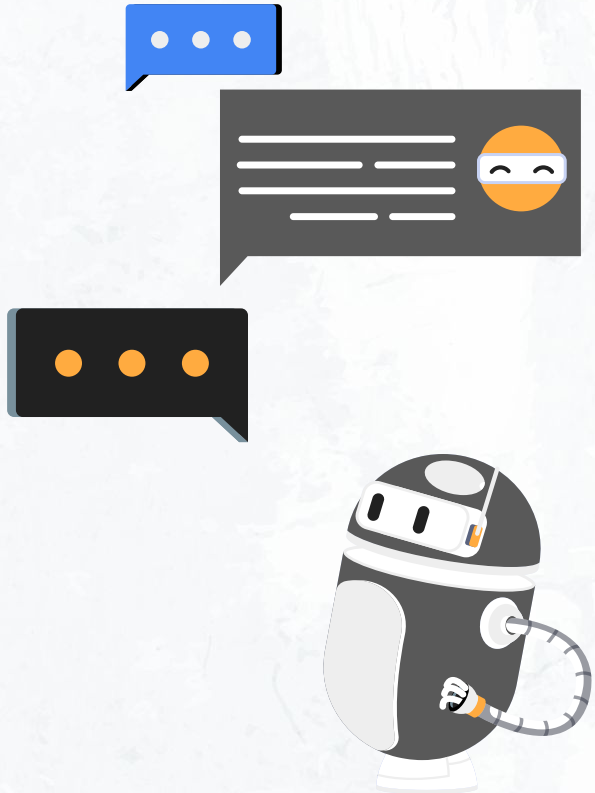
Sistem cu inteligență artificială pentru detecția automată a vorbirii înșelătoare



Echipa 8

Cuprins

01 → Introducere

Acest proiect abordează problema clasificării rostirilor în funcție de veridicitatea lor, folosind exclusiv semnalul audio. În particular, se urmărește construirea unui sistem automat de detecție a **vorbirii înșelătoare**, bazat pe caracteristici extrase algoritmic din segmente vocale, înregistrate într-un cadru controlat.

Pentru a evalua eficiența abordării, au fost testate și comparate mai multe tipuri de modele de clasificare:

- **SVM** (Support Vector Machines)
- **Random Forest (RF)**
- **Fully Connected Neural Network (FCNN)**
- **Convolutional Neural Network (CNN)**

02 → Setul de date

121 fișiere audio lungi:

- 61 fișiere etichetate ca „înșelătoare”
- 60 fișiere etichetate ca „sincere”

În urma segmentării, setul total cuprinde **929 rostiri**, dintre care:

- **463 sunt înșelătoare**,
- **466 sunt sincere**,
- provenind de la **56 de vorbitori unici:**

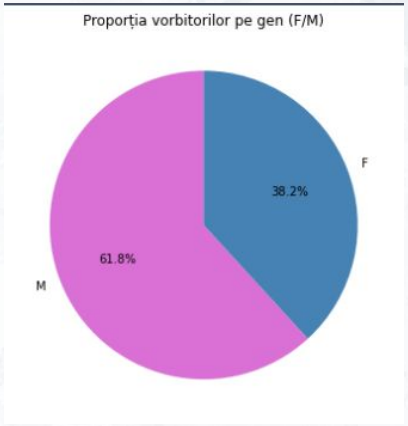
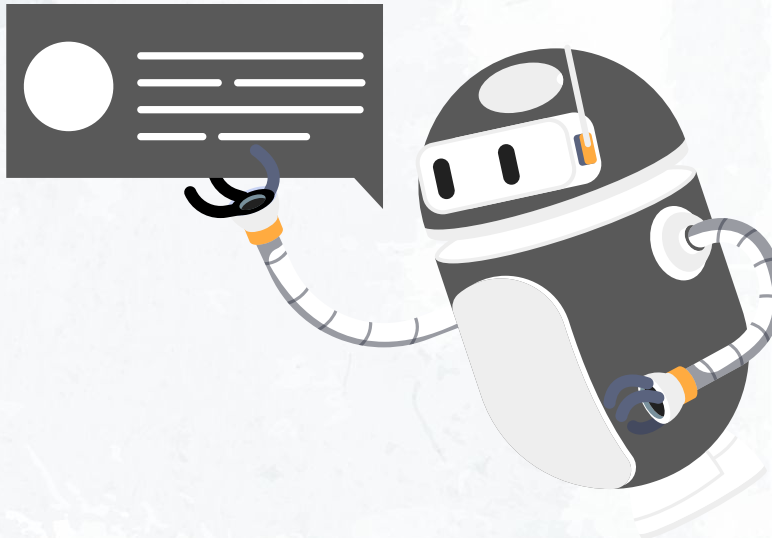


Fig.1 Proporția vorbitorilor pe gen



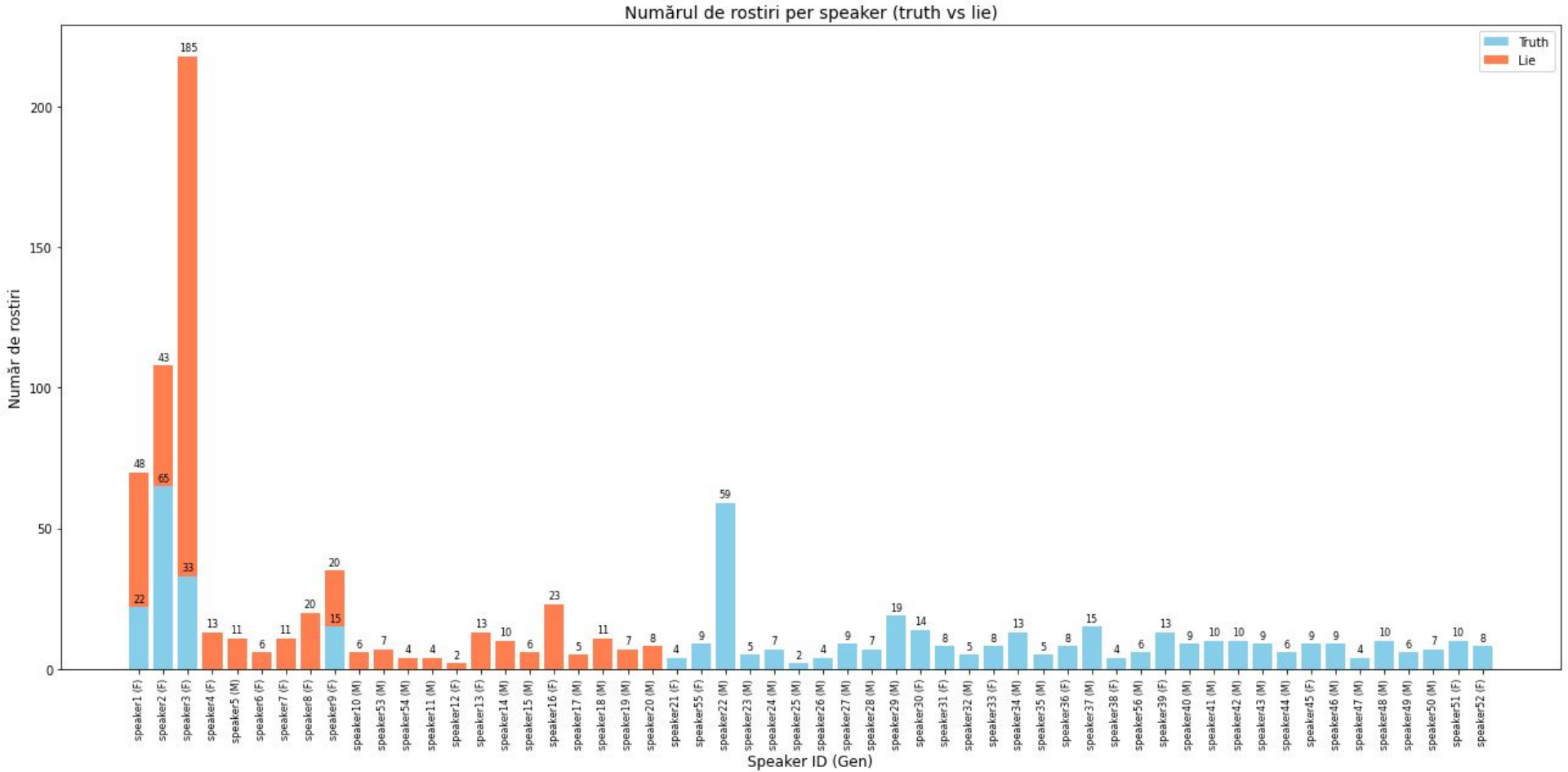


Fig.2 Numărul de rostiri per vorbitor

03 → Preprocesare și procesare date

Preprocesare

`extract_utternances.py`

- **Scop**: Segmentează înregistrările audio lungi în rostiri individuale.
- **Intrare**: fișiere `.wav` și `.csv` cu adnotări (moment început/sfârșit, ID, gen vorbitor).
- **Filtrare**: ignoră rostirile care nu aparțin subiecților (etichetați ca „TM”).
- **Ieșire**: fișiere `.wav` pentru fiecare rostire (929 rostiri).
- **Exemplu nume fișier**: `trial_lie_001_speaker03_utt5.wav`

Procesare

`extract_features.py`

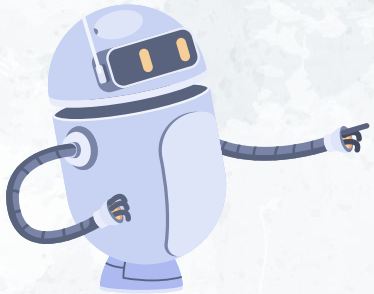
Extrage trăsături acustice algoritmice pentru clasificare (SVM / RF / FCNN)

Trăsături extrase (pe fiecare rostire):

- 13 coeficienți **MFCC**
- 13 coeficienți **Δ (delta MFCC)**
- 13 coeficienți **ΔΔ (delta-delta MFCC)**
- 1 coeficient **F0** (frecvență fundamentală)

Pentru fiecare: se calculează **media** și **deviația standard**

→ Total: **80 trăsături** per rostire



Etichetare și ieșire:

- `lie` → 1, `truth` → 0 (bazat pe numele fișierului)
- Ieșire: fișier CSV (`features.csv`) cu **929 instanțe**

Procesare

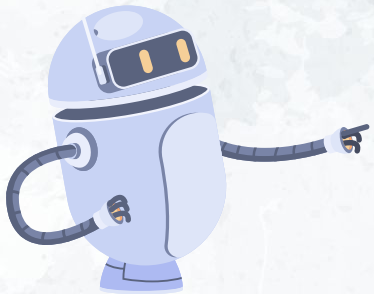
`normalize_features.py`

Aduce toate trăsăturile pe aceeași scală pentru a evita favorizarea unor dimensiuni î
clasificare.

Normalizare z-score per vorbitor:

- pentru fiecare vorbitor, se calculează:
  - media trăsăturii (μ),
  - deviația standard (σ),
- apoi, fiecare valoare este transformată astfel:

$$x_{norm} = \frac{x - \mu}{\sigma}$$



=> Fișier CSV cu trăsături **normalizate**: `features_normalized.csv`

Procesare

`extract_spectograms.py`

Cadrare și transformare

- **Frecvență eșantionare**: 16 kHz
- **Fereastră**: 25 ms (400 eșantioane), **pas**: 10 ms (160 eșantioane)
- **FFT**: 512 puncte
- **Fereastră analiză**: Hamming
- **Număr cadre**: 200 → acoperă **~2 secunde** de semnal
- **Zero-padding** pentru semnalele mai scurte
- **Conversie amplitudine**: log10(|X| + 1e-10), [dB]

Spectrograme liniare de intrare pentru CNN

- Frecvențe păstrate: 257 binuri (0–8 kHz)
- Dimensiune finală: 257 × 200
- Captură fidelă a detaliilor brute de frecvență
- Normalizare: z-score (media 0, deviație standard 1)

04 → Distribuția în fold-uri

`generate_folds_file.py`

1. **Generarea a 5 fold-uri echilibrate:**
   - •încarcă fișierul features.csv cu toate rostirile și adnotările.
   - •extrage genul fiecărui speaker din fișierele .csv de adnotări.
   - •calculează pentru fiecare speaker:
     - numărul de rostiri,
     - distribuția truth/lie,
     - genul.
   - •**sortează** descrescător speakerii după numărul de rostiri.
   - •**alocă fiecare speaker** într-unul din cele 5 fold-uri printr-un algoritm greedy care:
     - menține (pe cât posibil) echilibrul între genuri,
     - menține proporțiile între clase (truth/lie),
     - păstrează o dimensiune similară a datelor între folduri.

2. **Salvare și reutilizare**
   - •Fold-urile sunt salvate într-un fișier .pkl (pickle)
   - •Acest fișier conține, pentru fiecare din cele 5 fold-uri:
     - indexurile speakerilor folosiți pentru validare ,
     - restul indexurilor pentru antrenare .

3. **Utilizare consistentă în toate modelele:**
   Fiecare model (SVM, RF, FCNN, CNN) încarcă același fișier pickle pentru a asigura:
   - •consistență între experimente,
   - •comparații corecte ale performanțelor,
   - •independență completă între antrenare și validare.

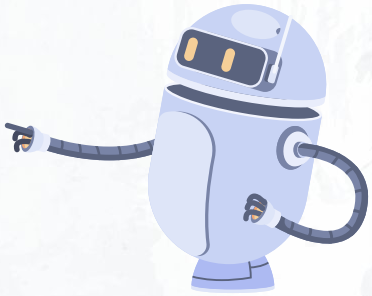
Distribuția în fold-uri

`generate_folds_file.py`

4. **Rezultat**

Fiecare fold conține:

- ~185–190 rostiri în validare (~20%)
- ~740 în antrenare (~80%)



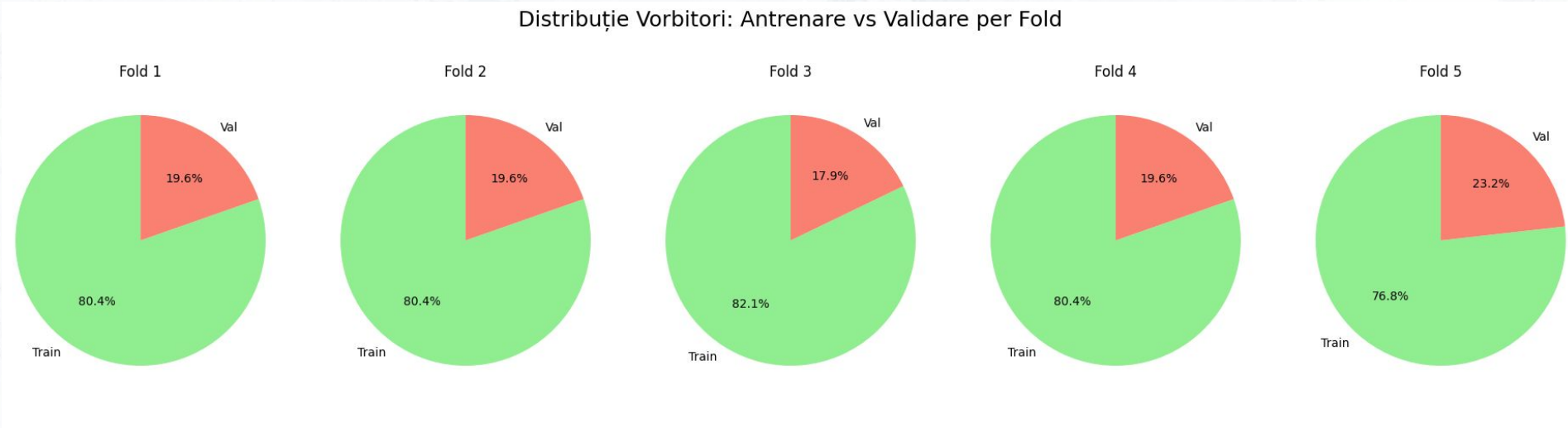


Fig.3 Distribuție vorbitori: antrenare vs validare

Distribuția în fold-uri

`generate_folds_file.py`

4.**Rezultat**

Se asigură:

✅ distribuție echilibrată pe gen (F vs. M)

✅ proporții apropiate adevăruri / minciuni

✅ vorbitori complet separați între antrenare și validare

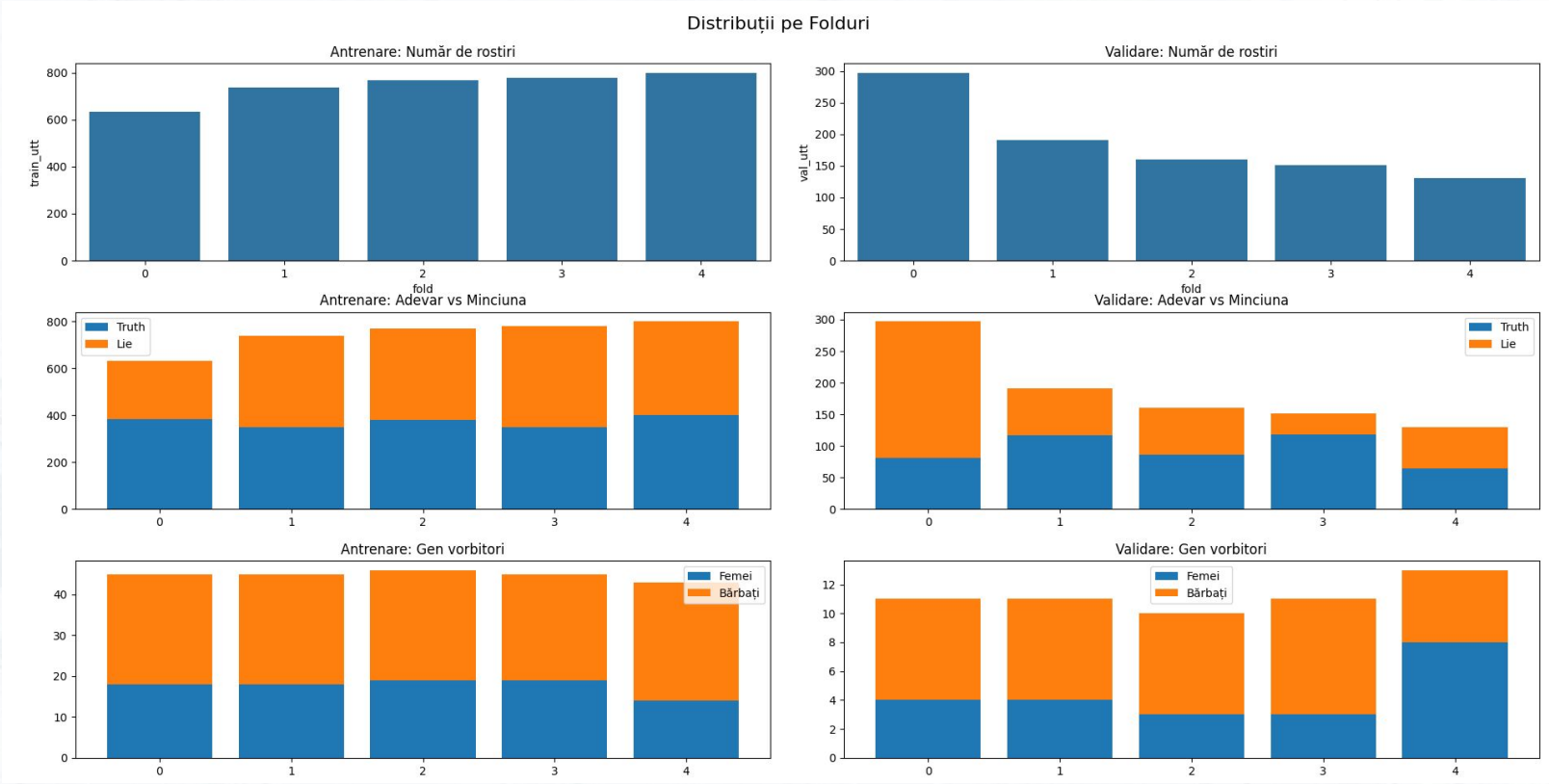


Fig.4 Distribuții pe fold-uri

Distribuția în fold-uri

`generate_folds_file.py`

Graficul ilustrează distribuția procentuală pe fiecare fold, separat pentru seturile de antrenare și validare, evidențiind echilibrul dintre clase (adevăr vs. minciună) și genul vorbitorilor (femei vs. bărbați), asigurând o împărțire echitabilă și o evaluare robustă a modelelor.

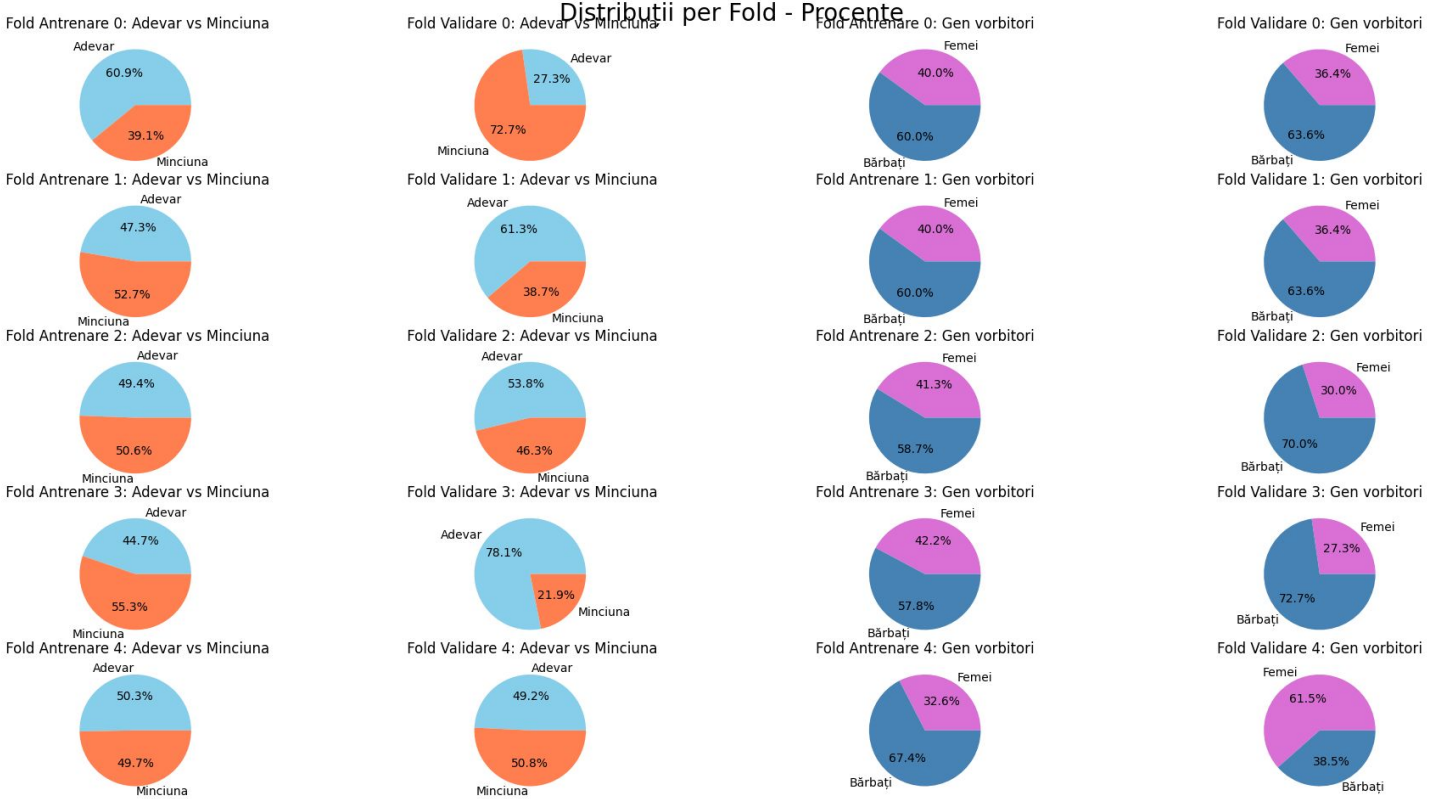


Fig.5 Distribuții per Fold

05 → Modele

SVM

Configurații totale testate: 10

Parametri variați:

- `C` (complexitate/penalizare): **0.1**, **1.0**, **5, 10.0**
- `kernel`: **rbf**, **linear**, **sigmoid**

Observații din grafic

- Configurația **`rbf (C=5.0)`** a obținut cele mai bune scoruri generale (atât la acuratețe cât și la F1).
- **Kernelul `sigmoid`** a avut performanțe mai scăzute în general.
- **Diferențele între acuratețe și F1** evidențiază dezechilibre în clase

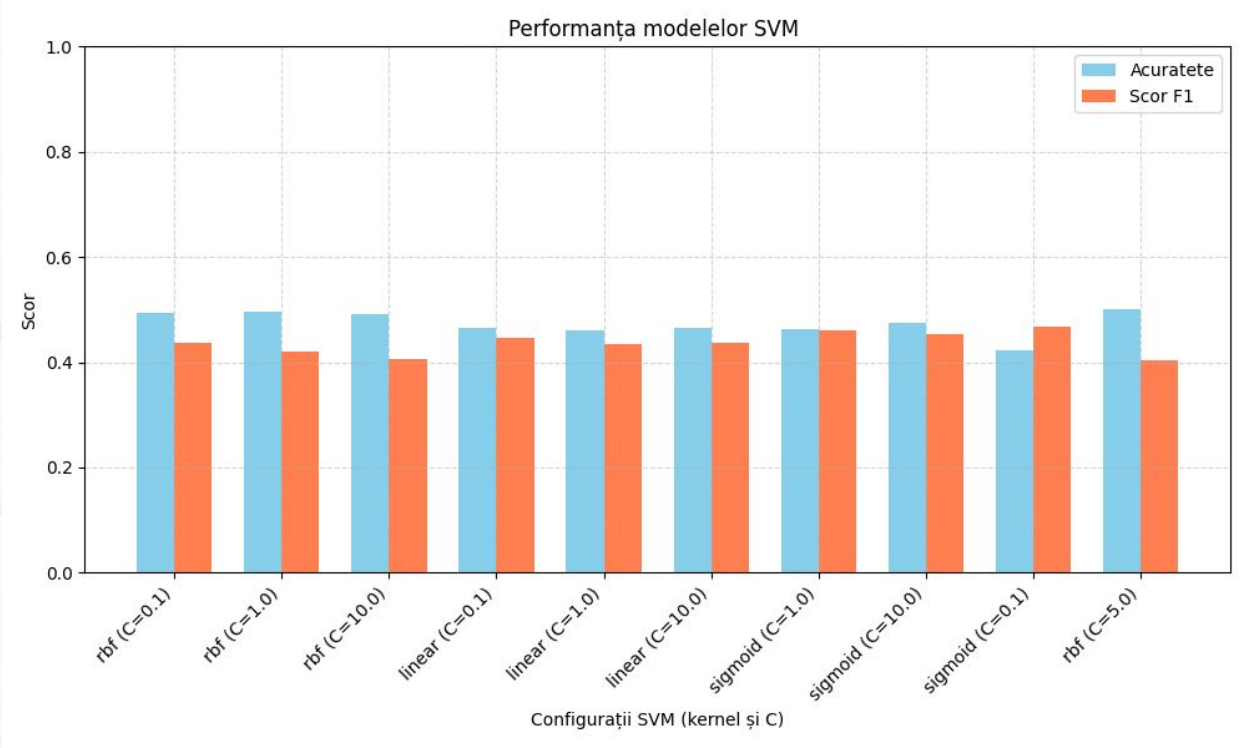


Fig.6 Performanța modelelor SVM

RANDOM FOREST

Am evaluat **13 configurații** RF (variații de `n_estimators-intre 40 si 200`, `max_depth intre 10 si 50`, `criterion-gini si entropy`)

- Configurații cu **aceiași parametri**, dar criteriu diferit (`gini` vs `entropy`), au arătat clar că **Gini a dus constant la rezultate mai bune**.
- `40x50_gini` vs `40x50_entropy` → diferență vizibilă de performanță, 48.14% vs. 43.66%
-
- Configurații cu **doar 40-60 estimatori** au oferit rezultate comparabile (sau mai bune) față de cele cu 100+ arbori.
- => Modelul poate rămâne **eficient computațional**, fără a sacrifica performanța.
- Valorile ridicate de `max_depth` (40–50) au permis captarea unor **modele mai complexe de minciună vs adevăr** în voce.

Cea mai performantă configurație:
`n_estimators = 40`, `max_depth = 50`, `criterion = 'gini'`

Fig.7. Performanța RF

RANDOM FOREST

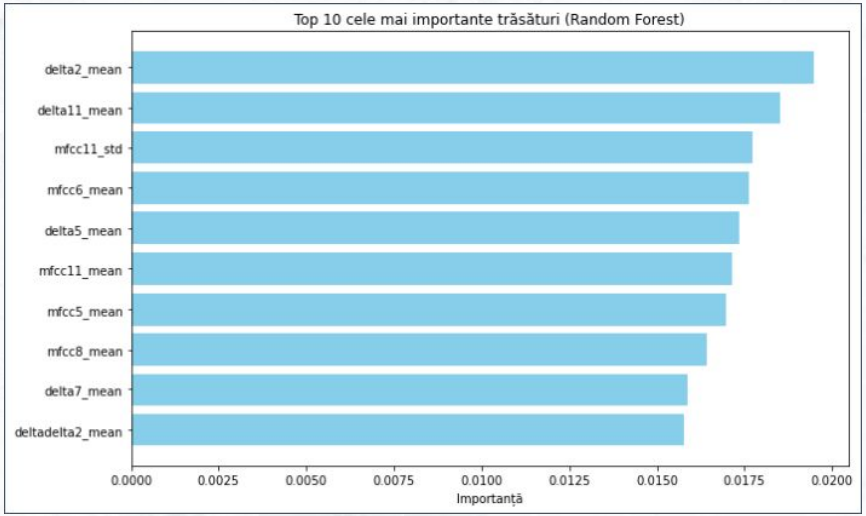


Fig.8 Top 10 trăsături RF

Importanța trăsăturilor a fost calculată pentru modelul Random Forest optim (40 arbori, adâncime 50). Este calculată ca contribuție medie la reducerea impurității în arborii de decizie. Valorile sunt relative (sumă totală = 1.0).

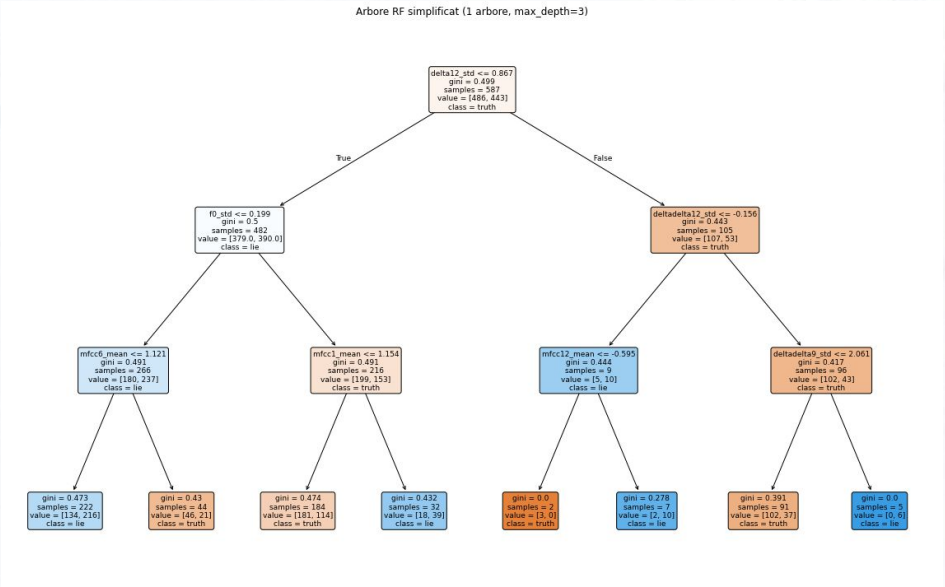


Fig.9 Arbore RF simplificat

Arbore demonstrativ. Pentru decizii reale, modelul folosește 40+ arbori cu adâncimi ≥20, combinând mii de astfel de impartiri.

FCNN

Configurații testate:

- **Număr total de configurații testate**: 100
- **Arhitecturi de straturi ascunse testate**:
  - 1 strat: `[64]`, `[128]`
  - 2 straturi: `[64, 32]`, `[128, 64]`
  - 3 straturi: `[128, 64, 32]`
- **Funcții de activare**:
  - ReLU
  - Tanh
- **Rate de dropout testate**:
  - 0.0 (fără dropout)
  - 0.2
  - 0.5
- **Regularizare L2**:
  - 0.0 (fără regularizare)
  - 0.001
- **Batch Normalization**:
  - Activat
  - Dezactivat
- **Funcție de pierdere**: Binary Cross-Entropy
- **Optimizator**: Adam
- **Early Stopping**:
  - Monitorizare: val_loss
  - Patience = 5 epoci
  - Restore best weights
- **Batch size**:
  - 32

Cele mai bune 5 configurații FCNN:

Nr.	Arhitectură (hidden_layers)	Activare	Dropout	L2	BatchNorm	Acuratețe	F1-score
1	[64, 32]	ReLU	0.5	0.0	Da	0.5593	0.4446
2	[128, 64]	ReLU	0.5	0.0	Da	0.5340	0.4749
3	[64]	ReLU	0.5	0.001	Da	0.5190	0.4615
4	[128, 64]	ReLU	0.2	0.001	Da	0.5023	0.4174
5	[128]	ReLU	0.2	0.0	Da	0.5003	0.4563

- Toate configurațiile folosesc activare **ReLU**.;
- Cele mai bune scoruri apar când se folosește **Batch Normalization** și **Dropout mare (0.5)**.
- Arhitecturile mai adânci, dar nu prea complexe (1–2 straturi), tind să funcționeze mai bine.

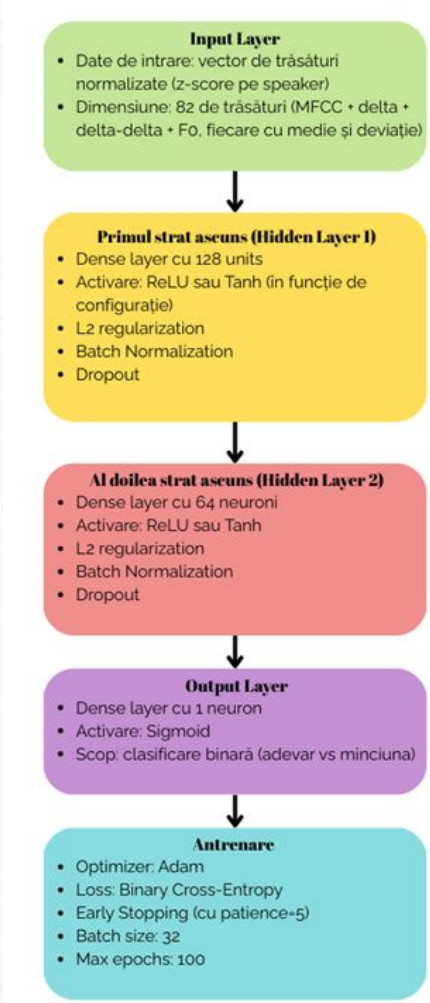


Fig. 10 Arhitectură FCNN

CNN

🟦 **Date de intrare:**
Spectrograme liniare (STFT, 257×200)
Toate normalizate, durată standardizată la ~2 secunde (200 cadre)

🟦 **Număr de configurații testate:** 55

🟦 **Parametri variați:**
- nr. filtre: 8, 16, 32, 64
- activare: ReLU, LeakyReLU, Tanh
- dense: 64, 128
- dropout: 0.3, 0.4
- LR: 1e-3, 5e-4, 2e-4

🟦 **Optimizator:** Adam |

🟦 **Early stopping:** Da

🟦 **Observații:**
- Cele mai bune rezultate cu ReLU + Dropout 0.3 + Batch Normalization
- Arhitecturile cu 2–3 straturi convoluționale + dense=64 au dat scorurile cele mai bune

Nr.	Filtre Conv	Activare Conv	Dense Units	Dropout	Acuratețe	F1-score
1	[8, 16]	ReLU	[64]	0.3	0.588	0.472
2	[8, 16]	LeakyReLU	[64]	0.3	0.571	0.408
3	[8, 16, 32]	ReLU	[64]	0.3	0.571	0.357
4	[8, 16]	LeakyReLU	[64, 64]	0.4	0.568	0.523
5	[8, 16]	LeakyReLU	[64, 64]	0.3	0.567	0.467

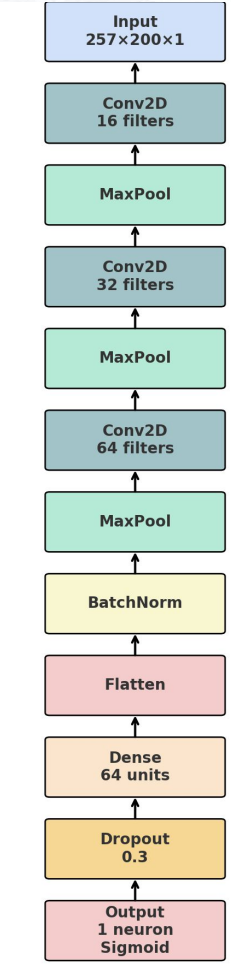


Fig. 11 Arhitectură CNN

Rezultate

- **CNN a obținut cea mai bună acuratețe**,datorită capacității de a extrage caracteristici complexe din spectrograme, precum ezitări, inflexiuni, pauze sau variații de tonalitate — tipare care pot indica vorbire înșelătoare.

- **FCNN** a avut performanțe superioare metodelor clasice, indicând utilitatea rețelelor neurale chiar și pe vectori de trăsături extrase manual.

- **SVM și RF** au performat sub 51%, sugerând că trăsăturile algoritmice (MFCC, F0) sunt insuficiente pentru separarea clară între vorbirea sinceră și cea înșelătoare.

- Diferențele de performanță arată că **modelele neurale sunt mai potrivite** pentru acest tip de problemă, unde variațiile subtile din voce pot fi capturate mai bine prin reprezentări spectrale detaliate.

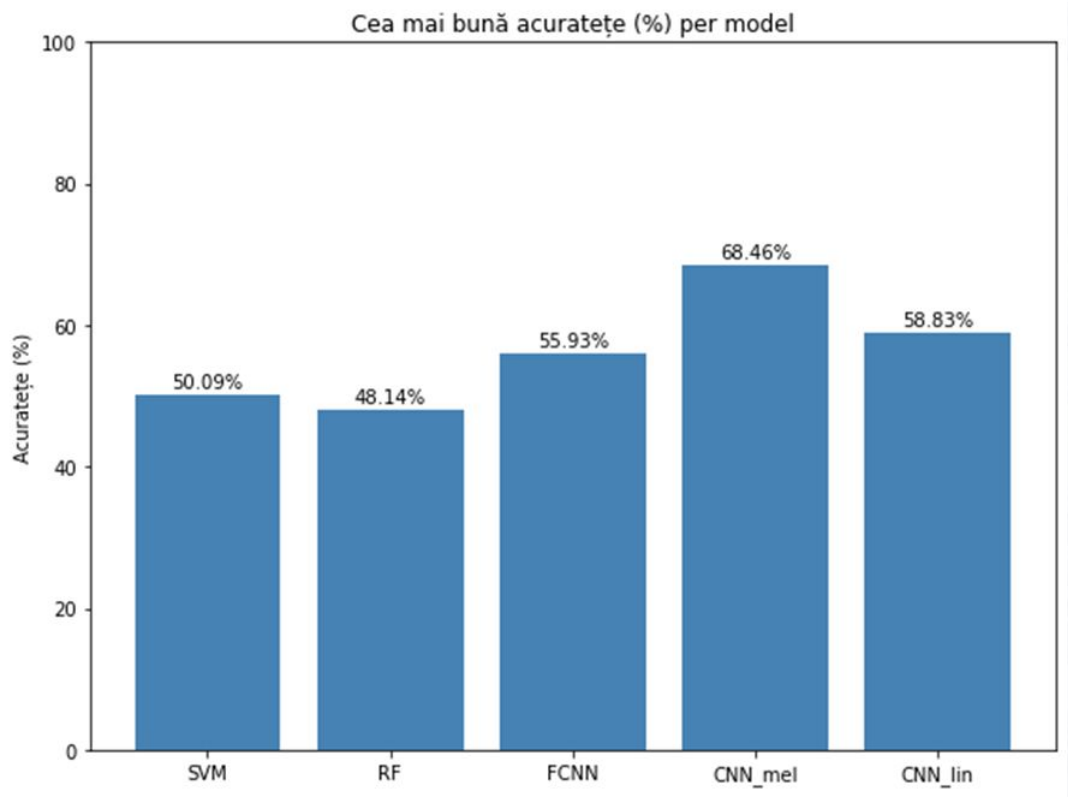


Fig. 12 Rezultate

06 → Concluzii

- **Datele sunt critice:** Segmentarea și echilibrarea (pe cat a fost posibil) setului de date (929 rostiri, 56 vorbitori) au asigurat evaluarea robustă.
- **CNN este optim:** Arhitectura CNN a oferit cea mai bună performanță, datorită capacității de procesare a semnalului brut învațând direct din reprezentări bogate în informație temporală și spectrală, spre deosebire de MFCC care comprimă datele.
- **Generalizare:** Folosirea fold-urilor echilibrate (80% testare, 20% validare, separare pe vorbitori) a prevenit overfitting-ul.

Limitări

- **Dimensiunea setului de date:** 929 de rostiri pot fi insuficiente pentru modele complexe.
- **Bias de gen:** *Setul de date a inclus 56 de vorbitori, dar distribuția pe gen nu a fost perfect echilibrată* (22F vs. 34M). *Deși am încercat să echilibrăm genul în fold-uri, mici discrepanțe în distribuția datelor pot introduce bias*
- **Lipsa diversității:** Unii vorbitori au avut doar una dintre clase (truth/lie), ceea ce poate introduce bias în antrenare.
- **Dezechilibru semnificativ:** Unii vorbitori au contribuit cu mult mai multe rostiri decât alții. Modelul poate învăța să recunoască vocea anumitor vorbitori (overfitting), nu trăsături generale ale minciunii.