# Proiect TAPI

## Tehnologii Avansate de Programare

## "Logistic Regression vs Random Forest - Classification Models"

**Student:** Brânzea Ana-Maria

Grupa 41312

# 1. DESCRIEREA PROBLEMEI

Proiectul are ca obiectiv dezvoltarea unor modele de clasificare pentru a prezice personajele din seria "Game of Thrones" mai sunt în viaţă sau nu. Setul de date utilizat conţine informaţii despre genul, titlurile de nobilime şi apariţiile personajelor în diferite cărţi ale seriei. Întrebarea principală este: putem prezice dacă un personaj va muri în funcţie de aceste caracteristici?

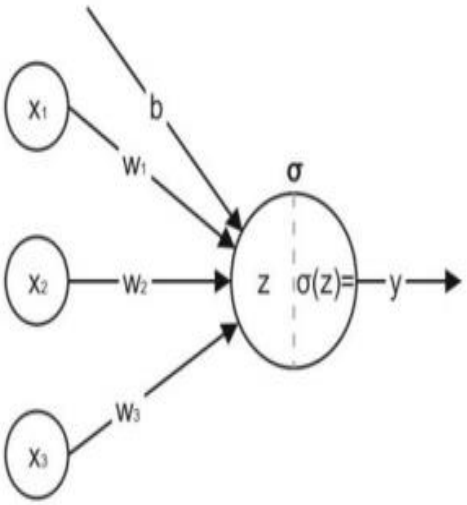
# 2. ELEMENTE TEORETICE PRIVIND CLASIFICATORII UTILIZAŢI ÎN PROIECT

## Logistic Regression

Regresia logistică este un **clasificator** utilizat în învăţarea supervizată. Ea estimează probabilitatea ca o variabilă dependentă binară (de exemplu, 0 sau 1, mort sau viu) să aibă o anumită valoare, bazată pe una sau mai multe variabile independente (atribute).

Regresia logistică a fost introdusă în 1958 de David Cox. Aceasta este utilizată în principal din două motive:

1. Oferă o interpretare a importanţei relative a atributelor setului de date.
2. Este de fapt o reţea neuronală cu un singur neuron, ceea ce face tranziţia la reţele neuronale şi deep learning uşoară.




Un model de regresie logistică tipic include următoarele componente:

- **Variabile de intrare (x_1, x_2, x_3):** Caracteristicile sau predictorii care influenţează rezultatul.
- **Greutăţi (w_1, w_2, w_3):** Coeficienţii care ponderizează importanţa fiecărei variabile de intrare.
- **Bias (b):** Un prag care ajustează intrarea netă.
- **Funcţia de activare (Funcţia sigmoid):** Transformă intrarea netă într-o probabilitate între 0 şi 1.

| ***FUNCȚIONARE*** | |
|---|---|
| *Calculul Logitului (Intrarea Netă)* | $z = b + w_1 * x_1 + w_2 * x_2 + w_3 * x_3$ (1) |
| *Funcția Sigmoid* | $\hat{y} = \varphi(z) = \frac{1}{1+e^{-z}}$ (2) |
| *Calculul Valorii Funcției de Ieșire* | $\hat{y} = \varphi(b + w_1 * x_1 + w_2 * x_2 + w_3 * x_3) = \frac{1}{1 + e^{-(b+w_1*x_1+w_2*x_2+w_3*x_3)}}$ |

SSE (Sum of Square Errors) este o metodă utilizată pentru a măsura eroarea totală a unui model. În contextul regresiei logistice, aceasta poate fi utilizată pentru a evalua cât de bine se potrivește modelul la datele de antrenament.

## CALCULUL SSE

SSE este calculată prin:

1. Calcularea erorii pentru fiecare punct de date, adică diferența dintre valoarea prezisă și valoarea reală.
2. Ridicarea erorii la pătrat pentru a se asigura că valorile negative nu se anulează cu cele pozitive.
3. Sumarea tuturor erorilor pătratice pentru a obține eroarea totală.

Formula SSE este: $E = \frac{1}{2}\sum_{i=1,n}\left(t^{(i)} - \hat{y}^{(i)}\right)^2$

Unde:

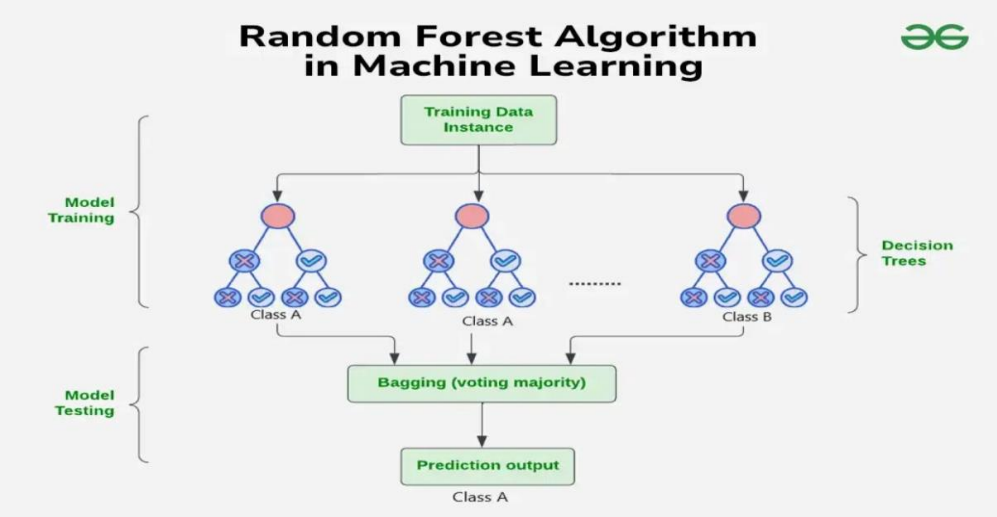
- $t^i$ este valoarea reală.
- $\hat{y}^i$este valoarea prezisă de model.
- $n$ este numărul total de observații.

În regresia logistică, scopul este de a minimiza SSE prin ajustarea greutăților și bias-ului. Un model bun va avea o valoare mică a SSE, indicând că predicțiile sunt foarte apropiate de valorile reale.

## Random forest

Random forest este un algoritm de învățare automată bazat pe ansambluri, utilizat atât pentru clasificare, cât și pentru regresie. Acesta construiește multiplii arbori de decizie în timpul antrenării și combină predicțiile acestora pentru a îmbunătăți acuratețea și robustețea modelului.
Random forest a fost introdus de Leo Breiman în 2001 și este utilizat pe scară largă în diverse domenii datorită performanței sale ridicate și a capacității de a gestiona date mari și complexe. Principalele avantaje ale utilizării random forest includ:

1. **Reducerea overfitting-ului:** Prin combinarea mai multor arbori, random forest reduce riscul de overfitting, care apare atunci când un model se potrivește prea bine pe datele de antrenament, dar eșuează pe datele noi.
2. **Robustețe la variabilele zgomotoase:** Random forest este capabil să gestioneze datele care conțin variabile nerelevante sau lipsă.




| ***FUNCȚIONARE*** | |
|---|---|
| *Crearea Subseturilor de Date* | Utilizează tehnicii de bootstrap pentru a crea multiple subseturi ale setului de date inițial. Fiecare arbore este antrenat pe un subset diferit de date, ceea ce face fiecare arbore unic. |
| *Construirea Copacilor de Decizie* | Fiecare arbore de decizie este construit utilizând o subset aleatoriu de date și de variabile. Acest proces reduce corelația dintre arbori și îmbunătățește diversitatea ansamblului, crescând astfel robustețea modelului final. |
| *Votarea Majoritară (Clasificare) sau Media (Regresie)* | După ce toți arborii de decizie sunt antrenați, random forest combină predicțiile acestora:<br>• **Clasificare:** Predicția finală este determinată prin votarea majoritară a predicțiilor individuale ale copacilor.<br>• **Regresie:** Predicția finală este media predicțiilor individuale ale copacilor. |

### ASUMȚIILE ALGORITMULUI RANDOM FOREST

- Fiecare arbore face propriile predicții independent de alții.
- Părți aleatorii ale datelor sunt utilizate pentru a construi fiecare arbore pentru a reduce greșelile.
- Este necesar un volum suficient de date pentru a asigura diversitatea arborilor și a învăța tipare unice.
- Combinarea predicțiilor din arbori diferiți conduce la rezultate finale mai precise.

## 3. SURSA DE DATE

Pentru acest proiect, am utilizat un fișier CSV intitulat character-deaths.csv (preluat de pe platforma Kaggle), care conține informații detaliate despre personajele din seria Game of Thrones. Datele sunt structurate în coloane specifice, fiecare oferind informații esențiale despre caracteristicile și evenimentele legate de personajele respective. Coloanele din acest fișier sunt următoarele:

- Name: Numele personajului.
- Allegiances: Casa de care aparține personajul.

- Death Year: Anul în care a murit personajul.
- Book of Death: Cartea în care a murit personajul.
- Death Chapter: Capitolul din cartea în care a murit personajul.
- Book Intro Chapter: Capitolul din carte în care a fost introdus personajul.
- Gender: Genul personajului (1 reprezintă masculin, 0 reprezintă feminin).
- Nobility: Statutul nobil al personajului (1 reprezintă nobil, 0 reprezintă om de rând).
- GoT: Apariţia personajului în prima carte (1 dacă a apărut, 0 dacă nu a apărut).
- CoK: Apariţia personajului în a doua carte (1 dacă a apărut, 0 dacă nu a apărut).
- SoS: Apariția personajului în a treia carte (1 dacă a apărut, 0 dacă nu a apărut).
- FfC: Apariţia personajului în a patra carte (1 dacă a apărut, 0 dacă nu a apărut).
- DwD: Apariţia personajului în a cincea carte (1 dacă a apărut, 0 dacă nu a apărut).

## Preprocesarea datelor

Analizând datele din csv, am observat că există anumite date lipsă pentru coloanele Allegiances, Death Year, Book of Death, Death Chapter, Book Intro Chapter, aşadar, le-am tratat astfel:

- Am eliminat duplicatele pentru a asigura unicitatea fiecărui personaj

```
got_prep=got.drop_duplicates()
```

- Am adăugat valoarea "unknown" pentru coloana 'Allegiances'

```
got_prep['Allegiances']=got_prep['Allegiances'].fillna('unknown')
```

- Am presupus că personajele care nu au un an al morţii Death Year, sunt în viaţă, deci am adăugat valoarea 0.

```
got_prep['Death Year']=got_prep['Death Year'].fillna(0)
got_prep['Death Year']=got_prep['Death Year'].astype('int')
```

- Pentru 'Book of Death' şi 'Death Chapter' am adăugat valori diferite (valori de -1) pentru a indică că personajele sunt încă în viaţă

```
got_prep['Book of Death']=got_prep['Book of Death'].fillna('-1')
got_prep['Death Chapter']=got_prep['Death Chapter'].fillna('-1')
```

- Am eliminat coloana Book Intro Chapter pentru că nu este folositoare.

```
got_prep=got_prep.drop(['Book Intro Chapter'], axis=1)
```

## Împărţirea Setului de Date

Pentru a evalua performanţa modelului, am împărţit setul de date în seturi de antrenament şi de testare. Setul de antrenare X include coloanele **Gender, Nobility, GoT, CoK, SoS, FfC şi DwD**, iar setul de testare y indică dacă un personaj este mort (1) sau viu (0). Aceste seturi au fost împărţite în proporţiile 70% pentru antrenare şi 30% pentru testare, asigurând astfel o evaluare corectă şi fiabilă a performanţei modelului.

```
X=got_prep[['Gender', 'Nobility', 'GoT', 'CoK', 'SoS', 'FfC', 'DwD']]
y=(got_prep['Death Year']>0).astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

# 4. MODELE DE ML IMPLEMENTATE

## Logistic Regression

Modelul estimează probabilitatea ca un personaj să fie mort (1) sau viu (0). Parametrii utilizați în acesta sunt:

- C: Parametru de regularizare care controlează forța regularizării. Valori mai mici specifică o regularizare mai puternică.
- max_iter: Numărul maxim de iterații pentru algoritmul de optimizare.
- random_state: Asigură reproducibilitatea rezultatelor prin setarea unei stări aleatorii fixe.

### CODUL SURSĂ

```
#1.Logistic regression

# Antrenarea modelului de regresie logistică
model_lr=LogisticRegression(C=0.2, max_iter=1000,random_state=42)
model_lr.fit(X_train, y_train)

# Prezicerea valorilor pentru setul de testare
y_pred_lr=model_lr.predict(X_test)

# Evaluarea performanței modelului
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print(f'Acuratetea modelului LR: {accuracy_lr}')
```

### PERFORMANȚA MODELULUI

| C | Max_iter | Acuratețe |
|---|---|---|
| 0,01 | 100 | 0,717 |
| 0,1 | 100 | 0,713 |
| 0,1 | 1000 | 0,713 |
| 1 | 500 | 0,735 |
| **10** | **100** | **0.735** |
| 1000 | 100 | 0,735 |

**Observații:**

- **n_estimators**: Numărul de estimatori nu pare să aibă un impact consistent asupra acurateții. De exemplu, n_estimators de 10 cu max_depth de 10 sau 100 obține acurateți similare (0.731).

- **max_depth**: Adâncimea maximă a copacilor are un impact variabil asupra performanței. Se observă că un max_depth de 100 sau None nu aduce o îmbunătățire consistentă.
- **Acuratețea Maximă**: Cel mai bun rezultat, o acuratețe de 0.735, este obținut cu n_estimators de 8 și max_depth de 100.

## Random Forest

Parametrii utilizați în acest model sunt:

- n_estimators: Numărul de arbori.
- max_depth: Adâncimea maximă a fiecărui arbore.
- random_state: Asigură reproducibilitatea rezultatelor prin setarea unei stări aleatorii fixe.

## COD SURSĂ

```
#2. Random Forest
from sklearn.ensemble import RandomForestClassifier
# Antrenarea modelului Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, max_depth=100)
rf_model.fit(X_train, y_train)
# Prezicerea valorilor pentru setul de testare
y_pred_rf = rf_model.predict(X_test)

# Evaluarea performantei modelului
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Acuratetea modelului Random Forest: {accuracy_rf}')
```

## PERFORMANȚA MODELULUI

| n_estimators | max_depth | Acuratețe |
|---|---|---|
| 10 | 10 | 0,731 |
| **8** | **100** | **0,735** |
| 10 | 100 | 0,731 |
| 50 | 10 | 0,713 |
| 50 | 100 | 0,713 |
| 100 | 10 | 0.706 |
| 1000 | none | 0,721 |

**Observații:**

- **Valorile lui C**: Acuratețea modelelor crește odată cu creșterea valorii lui C. Se observă că C mai mari (1, 10, 1000) duc la acurateți mai mari (0.735).
- **Max_iter**: Variațiile valorii Max_iter între 100 și 1000 nu par să aibă un impact semnificativ asupra acurateții modelului.
- **Acuratețea Maximă**: Cel mai bun rezultat, o acuratețe de 0.735, este obținut cu valori ale lui C de 1, 10 și 1000, indiferent de Max_iter.

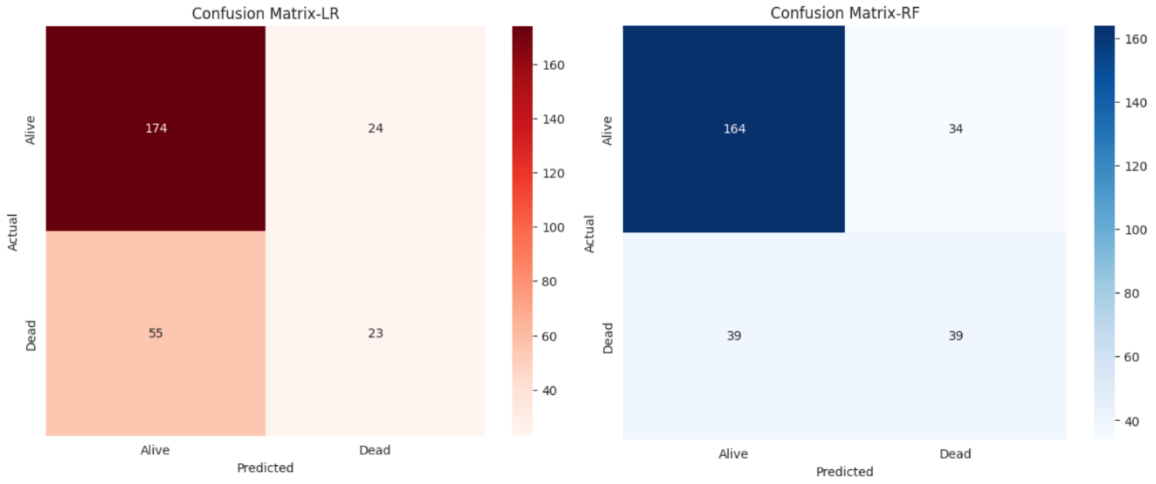
# 5. ANALIZA COMPARATIVĂ MODELE ML ŞI EXPLICAŢII PREDICŢII

Pentru a compara performanţa modelelor de ML, vom utiliza metrica acurateţei şi vom analiza matricile de confuzie generate pentru fiecare model. Acurateţea reprezintă proporţia predicţiilor corecte realizate de model faţă de totalul predicţiilor şi este o măsură importantă pentru evaluarea performanţei modelului.

În urma experimentelor realizate, cele mai bune rezultate pentru modelele Logistic Regression şi Random Forest au fost obţinute astfel:

| **Logistic Regression:** | **Random Forest:** |
|---|---|
| C: 1, 10, 1000<br>Max_iter: 100, 500, 1000 | n_estimators: 8<br>max_depth: 100 |
| Acurateţe: **0.735** | **Acurateţe: 0.735** |

De asemenea, matricele de confuzie permit evaluarea mai detaliată despre performanţa fiecărui model, indicând numărul de predicţii corecte şi incorecte pentru fiecare clasă. Mai jos se găsesc matricele pentru fiecare ML:




Modelul de Logistic Regression se descurcă mai bine la prezicerea personajelor vii (precizie de 79%) comparativ cu cele moarte (precizie de 54%). Recall-ul pentru prezicerea personajelor moarte este de 40%, ceea ce sugerează că modelul este mai conservator în prezicerea morţii. În general, modelul arată o performanţă decentă, dar ar putea beneficia de ajustări suplimentare şi poate de adăugarea unor caracteristici suplimentare pentru a-şi îmbunătăţi capacitatea predictivă, în special pentru personajele moarte.

```
report = classification_report(y_test, y_pred_lr, target_names=['Alive', 'Dead'])

(accuracy_lr, report)
```

```
(0.7355072463768116,
 '              precision    recall  f1-score   support\n\n       Alive       0.79      0.87      0.82
198\n        Dead       0.54      0.40      0.46        78\n\n    accuracy                           0.74
276\n   macro avg       0.66      0.63      0.64       276\nweighted avg       0.72      0.74      0.72
276\n')
```

Modelul Random Forest este mai precis în prezicerea personajelor vii (precizie de 81%) şi are un recall ridicat pentru acestea (83%), ceea ce indică faptul că modelul este bun în identificarea personajelor vii. Pentru personajele moarte, modelul are o precizie decentă (53%) şi un recall moderat (50%), sugerând că modelul are o performanţă echilibrată între prezicerea personajelor vii şi moarte.

```
report = classification_report(y_test, y_pred_rf, target_names=['Alive', 'Dead'])

(accuracy_rf, report)
```

```
(0.7355072463768116,
 '              precision    recall  f1-score   support\n\n       Alive       0.81      0.83      0.82
198\n        Dead       0.53      0.50      0.52        78\n\n    accuracy                           0.74
276\n   macro avg       0.67      0.66      0.67       276\nweighted avg       0.73      0.74      0.73
276\n')
```

În concluzie, ambele modele prezintă puncte forte şi slăbiciuni. Logistic Regression tinde să fie mai echilibrat între prezicerea personajelor vii şi moarte, în timp ce Random Forest excelează în prezicerea personajelor vii şi menţine un echilibru mai bun pentru personajele moarte.

## LIME

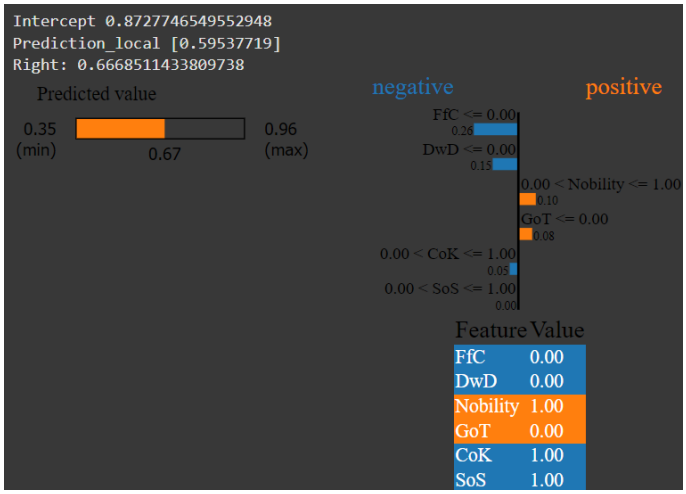
Pentru a înţelege mai bine predicţiile făcute de modelul Logistic Regression, am utilizat LIME (Local Interpretable Model-agnostic Explanations). LIME creează modele interpretabile local în jurul fiecărei predicţii pentru a oferi informaţii despre factorii care au influenţat decizia modelului.

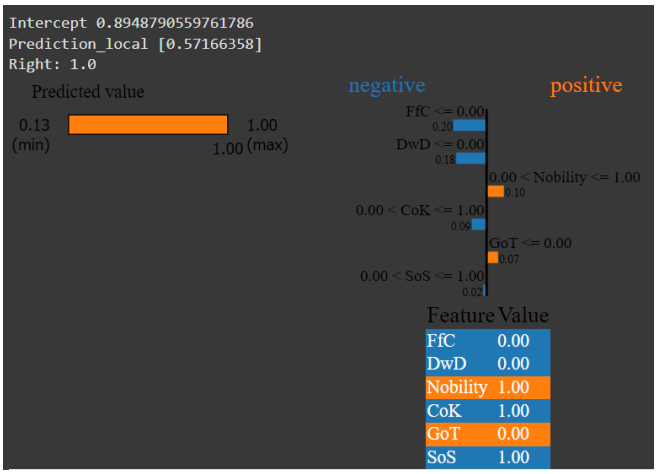
```
# LIME has one explainer for all the models
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_names=X_train.columns.values.tolist(),
                                                   class_names=['Death Year'], verbose=True, mode='regression')
j=10
X_train.columns.values.tolist()
X_test_df = pd.DataFrame(X_test.values, columns=X.columns)
```

S-a creat unui explicator LIME() pentru toate modelele, utilizând valorile setului de date de antrenament şi specificând numele caracteristicilor din setul de date. Explicatorul LIME este configurat pentru a funcţiona în modul de regresie.
Se selectează un eşantion pentru explicaţii, folosind indexul j  pentru a alege eşantionul din setul de testare. Lista coloanelor caracteristicilor este utilizată pentru a crea un DataFrame pentru setul de testare, utilizând valorile şi coloanele corespunzătoare.

**Pentru j=5:**

Logistic Regression- Valoarea prezisă este 0.67. Variabilele **FfC**, **DwD, CoK** şi **SoS** au o influenţă negativă asupra predicţiei, în timp ce variabilele **Nobility**, **GoT** au o influenţă pozitivă asupra predicţiei.

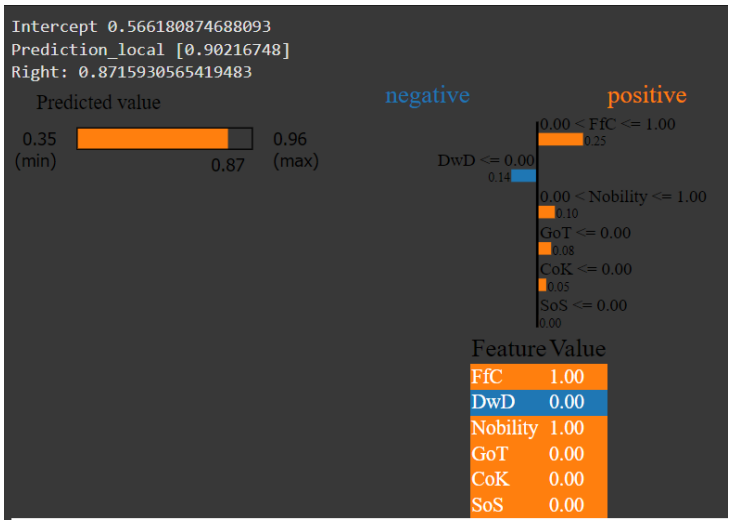



Random Forest- Valoarea prezisă este 1.00. Variabilele **Nobility**, **GoT** au o influenţă pozitivă asupra predicţiei, în timp ce variabilele **FfC**, **DwD, CoK** şi **SoS** au o influenţă negativă.

**Pentru j=10:**

Logistic Regression- valoarea prezisă este 0.87. Variabilele **FfC** şi **Nobility** au o influenţă pozitivă, în timp ce variabila **DwD** are o influenţă negativă asupra predicţiei. Valorile celorlalte variabile (GoT, CoK, SoS) sunt nule, indicând că nu au influenţă asupra predicţiei.



# 6. CONCLUZII

În urma analizei, am constatat că modelele Logistic Regression şi Random Forest au demonstrat capacitatea de a clasifica personajele din "Game of Thrones" pe baza probabilităţii de supravieţuire. Random Forest s-a remarcat printr-o performanţă superioară în ceea ce priveşte precizia şi recall-ul, fiind mai eficient în predicţiile corecte. Pe de altă parte, Logistic Regression a oferit un echilibru mai bun între clase, având o abilitate mai bună de a gestiona distribuţia inegală între clasele majoritare şi minoritare.

Scorul maxim obţinut, de 73,5%, indică faptul că modelele pot face predicţii relevante, dar există însă oportunităţi de îmbunătăţire. Adăugarea unor noi caracteristici, precum relaţiile dintre personaje, statutul social sau alte detalii relevante din poveste, ar putea creşte semnificativ performanţa modelelor. Totodată, explorarea altor algoritmi de învăţare automată, precum XGBoost,KNN, Gradient Boosting ar putea oferi perspective valoroase şi rezultate mai bune.

De asemenea, utilizarea LIME pentru analiza explicativă a evidenţiat importanţa variabilelor Nobility şi GoT în deciziile modelelor, subliniind că aceste atribute joacă un rol esenţial în predicţii. Acest tip de interpretabilitate este crucial pentru înțelegerea procesului decizional şi pentru ajustarea modelelor în funcţie de context.

În concluzie, ambele modele analizate sunt utile pentru această sarcină, însă alegerea lor depinde de nevoile specifice ale aplicaţiei. În plus, prin rafinarea dataset-ului şi testarea unor metode mai complexe, potențialul analizei poate fi extins.

# Bibliografie


1. https://www.kaggle.com/datasets/mylesoneill/game-of-thrones
2. https://timf.upg-ploiesti.ro/cursuri/cursuri/TAIA/TAIA_2024.pdf
3. https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html
4. https://scikit-learn.org/1.6/modules/generated/sklearn.ensemble.RandomForestClassifier.html
5. Sebastian Raschka & Vahid Mirjalili, Python Machine Learning. Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow2, 3rd Edition, 2019 – Biblioteca ITIMF
6. Neural Networks and Deep Learning, Michael Nielsen, The original online book can be found at http://neuralnetworksanddeeplearning.com
7. Python Machine Learning By Example: The easiest way to get into machine learning, Yuxi (Hayden) Liu, Packt Publishing, May 31, 2017 – Biblioteca ITIMF
8. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/
9. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/
10. https://www.kaggle.com/code/prashant111/explain-your-model-predictions-with-lime