



UNIVERSITATEA DIN
BUCUREȘTI

FACULTATEA DE
MATEMATICĂ ȘI
INFORMATICĂ



SPECIALIZAREA MATEMATICĂ-INFORMATICĂ

Lucrare de licență

IDENTIFICAREA PACIENȚILOR DIAGNOSTICAȚI CU COVID-19 FOLOSIND ÎNVĂȚAREA AUTOMATĂ

Absolvent

Elena Olaru

Coordonator științific

Prof. Dr. Florentin Ipată

București, iunie 2022

Rezumat

Începând din 2019, pandemia globală a fost în curs de desfășurare și în 2022 încă afectează tot mai multe persoane.

În această lucrare am ales să dezvolt mai multe modele de predicție, bazate pe tehnici de Data Mining și Machine Learning, pentru a veni în ajutorul combaterii virusului, prin diagnosticarea mai ușoară pe baza simptomelor. Doresc să urmăresc mai multe metode de învățare automată a nucleului de rezolvare a problemelor de clasificare, regresie și grupare în clustere. Acestea au performanțe bune, dar și unele limitări. Scopul este observarea și analizarea fiecărei metode, precum și a avantajelor și a dezavantajelor ce stau la baza implementării și a funcționării lor, pentru a găsi cel mai optim algoritm pentru a rezolva problema diagnosticării.

Abstract

Since 2019, the global pandemic has been ongoing and in 2022 it is still affecting more and more people.

In this thesis, I chose to develop several prediction templates based on Data Mining and Machine Learning techniques, so as to aid in tackling the virus through a simpler symptom based diagnosis. I am aiming to follow several Machine Learning methods, that will solve the core classification issues, regressions and clustering. These have a prominent performance but they also present some limitations. The main objective is the observation and analysis of each method, their advantages and disadvantages in order to find the most adequate algorithm to solve the problem of diagnosis.

Cuprins

1	Introducere	5
1.1	Problema abordată	5
1.2	Scopul lucrării	5
1.3	Obiective	6
1.4	Structura lucrării	6
1.5	Motivația personală	6
2	Preliminarii	7
2.1	Cerințele și provocările extragerii de date	7
2.2	Setarea datelor	7
2.3	Preprocesarea datelor	10
2.4	Procesul de învățare automată	10
3	Tehnici abordate	12
3.1	Alegerea limbajului de programare	12
3.2	Metrici de performanță	12
3.2.1	Acuratețea	13
3.2.2	Sensibilitatea	13
3.2.3	Specificitatea	14
3.3	Implementare	16
3.4	Configurarea algoritmilor	16
3.5	Evaluarea modelelor	18
3.6	Rezultate comparative	18
3.7	Determinarea celor mai importante caracteristici	23
3.8	Configurarea unei interfețe grafice	24
3.8.1	Tehnologii folosite	24
3.8.2	Modul de funcționare	24
4	Algoritmi utilizați	26
4.1	Mașina cu suport vectorial	26
4.1.1	Avantaje	26

4.1.2	Provocări	27
4.2	Arborele de decizie	27
4.2.1	Avantaje	28
4.2.2	Provocări	28
4.3	Cei mai apropiați K vecini (K-nearest neighbors)	30
4.3.1	Avantaje	30
4.3.2	Provocări	30
4.4	Random Forest	33
4.4.1	Avantaje	33
4.4.2	Provocări	33
4.5	Regresia logistică	35
4.5.1	Avantaje	35
4.5.2	Provocări	35
5	Concluzii	37
	Bibliografie	38

Capitolul 1

Introducere

1.1 Problema abordată

Noua pandemie de coronavirus (COVID-19), cauzată de virusul SARS-CoV-2, reprezintă în continuare o amenințare critică și urgentă pentru sănătatea globală [1]. Focarul de la începutul lunii decembrie 2019 din provincia Hubei din Republica Populară Chineză s-a răspândit în întreaga lume. În octombrie 2020, numărul total de pacienți confirmați că au boala a depășit 39.500.000, în mai mult de 180 de țări, deși numărul persoanelor infectate poate fi mult mai mare [2].

Această pandemie continuă să provoace sistemele medicale din întreaga lume în multe aspecte, inclusiv creșterea bruscă a cererilor de paturi de spital și lipsa critică de echipamente medicale, în timp ce mulți lucrători din domeniul sănătății au fost ei înșiși infectați. Astfel, capacitatea de a lua decizii clinice imediate și utilizarea eficientă a resurselor de asistență medicală este crucială. Cel mai validat test de diagnostic pentru COVID-19, folosind reacția polimerazei cu transcriptază inversă (RT-PCR), a lipsit de mult timp în țările în curs de dezvoltare. Acest lucru contribuie la creșterea ratelor de infecție și întârzie măsurile preventive critice.

1.2 Scopul lucrării

Metodele descrise în această lucrare își propun să vină în ajutorul combaterii pandemiei, întrucât permit diagnosticarea rapidă și eficientă a COVID-19 și pot atenua sarcina asupra sistemelor de sănătate. Au fost dezvoltate modele de predicție care combină mai multe caracteristici pentru a estima riscul de infecție, în speranța de a asista personalul medical din întreaga lume în triajul pacienților, mai ales în contextul resurselor limitate de asistență medicală. Predicția se realizează folosind informațiile clinice ale pacienților. Scopul este de a găsi algoritmul optim care poate identifica dacă un pacient este diagnosticat cu COVID-19.

1.3 Obiective

În implementarea lucrării am avut în vedere următoarele obiective:

- **Eficiență:** Pregătirea unui model de învățare automată care ar putea face predicții cât mai precise despre pacienții diagnosticați cu COVID-19 .
- **Antrenament:** Identificarea celei mai potrivite tehnici de învățare automată pentru predicție, pentru a se efectua pe rapoartele clinice ale pacienților.
- **Analiză:** Identificarea caracteristicilor care afectează predicția COVID-19 la pacienți.

1.4 Structura lucrării

Lucrarea se împarte în următoarele capitole:

1. **Introducere:** oferă o perspectivă a acestei lucrări, urmărind scopul, obiectivele, motivația.
2. **Preliminarii:** descrie pașii care se urmează în procesarea datelor și contextul învățării automate.
3. **Tehnici abordate:** urmărește configurarea și evaluarea metodelor de predicție.
4. **Algoritmi utilizați:** prezintă o analiză a algoritmilor de învățare prezenți.
5. **Concluzii:** determină algoritmului optim, rezumând informațiile prezentate.

1.5 Motivația personală

Am ales această temă deoarece vreau sa ofer sprijin cadrelor medicale în combaterea pandemiei, urmărind principiile de Data Mining și Machine Learning pentru a aborda cea mai optimă soluție.

Fiind un domeniu nou pentru mine, cercetarea pe aceste ramuri mi-a oferit o bună perspectivă asupra algoritmilor de învățare automată, privind eficiența și stabilitatea acestora în domeniul medical.

Capitolul 2

Preliminarii

2.1 Cerințele și provocările extragerii de date

Extragerea de date („*Data Mining*”) se referă la procesul de extragere a informațiilor utile din seturi mari de date. Aceasta poate fi aplicată și în cazul clasificărilor, prefigurându-se o serie de cerințe și trăsături, după cum urmează:

1. **Eficiența**, în cazul algoritmilor de extragere de date.
2. **Exactitatea și utilitatea rezultatelor**. Rezultatele descoperite necesită să reflecte cu exactitate conținutul bazei de date, pentru a fi utile în cercetări și aplicații practice. În cazul imperfecțiunilor, acestea sunt tratate ca măsuri de incertitudine, prin diferite metode, care vor fi enumerate în secțiunea 2.3.
3. **Modul de exprimare a rezultatelor, în cazul tipurilor variate de date**. După cum spune și definiția, extragerea se realizează din seturi mari de date, din care pot fi scoase tipuri diferite de cunoștințe. Analiza acestora se face și este prezentată din mai multe puncte de vedere, astfel că este nevoie de un mod expresiv de exprimare a rezultatelor, pentru ca rezultatele descoperite să poată fi înțelese și utilizate cu ușurință de toată lumea.

2.2 Setarea datelor

Colectarea datelor constituie cel mai important proces în această lucrare. Indiferent de domeniul de cercetare, acuratețea colectării datelor este esențială pentru menținerea coeziunii.

Ministerul Israelian al Sănătății a publicat în mod public date despre persoanele care au fost testate pentru SARS-CoV-2 prin testul RT-PCR [3].

Pe baza acestor date, am dezvoltat un model care prezice rezultatele testelor COVID-19 folosind opt caracteristici binare: sex, vârsta de 60 de ani sau peste, contact cunoscut cu o persoană infectată și cinci simptome clinice inițiale.

1. Informații de bază

- (a) Sex (bărbat/femeie)
- (b) Varsta ≥ 60 ani (adevărat/fals)

2. Simptome

- (a) Tuse (adevărat/fals)
- (b) Febra (adevărat/fals)
- (c) Durere de gât (adevărat/fals)
- (d) Dificultate în respirație (adevărat/fals)
- (e) Durere de cap (adevărat/fals)

3. Alte informații

- (a) Contact cunoscut cu o persoană confirmată că are COVID-19 (adevărat/fals)

Pentru a urmări o analiză cât mai concretă, am ales ca seturile pe care modelele vor fi antrenate să conțină înregistrări de la 400, 500, 1000, 1500, respectiv 2000 de teste individuale (dintre care 9, 15, 24, 42, respectiv 62 au fost confirmați că au COVID-19), din perioada 22 martie 2020 până în 31 martie 2020. Prin variația seturilor de date pot observa mai bine cum evoluează modelele.

Caracteristicile setului de date utilizat în această lucrare sunt evidențiate în tabelul din figura 2.1.

Caracteristică		400 înregistrări	500 înregistrări	1000 înregistrări
Sex	femeie	210	268	538
	bărbat	190	232	462
Varsta 60 ani	adevărat	60	75	142
	fals	340	425	858
Tuse	adevărat	36	36	38
	fals	364	464	962
Febră	adevărat	15	15	17
	fals	385	485	983
Durere de gât	adevărat	3	3	5
	fals	397	497	995
Dificultate în respirație	adevărat	1	1	1
	fals	399	499	999
Durere de cap	adevărat	1	2	3
	fals	399	498	997
Contact cu o persoană confirmată	adevărat	20	21	25
	fals	380	479	975

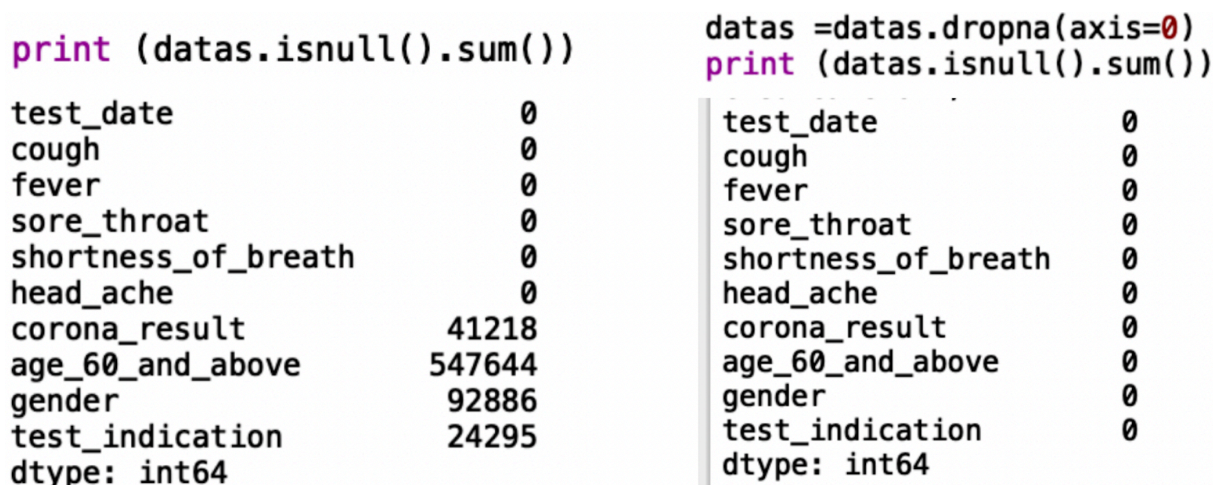
Caracteristică		1500 înregistrări	2000 înregistrări
Sex	femeie	807	1074
	bărbat	693	926
Varsta 60 ani	adevărat	205	277
	fals	1290	1713
Tuse	adevărat	41	47
	fals	1459	1953
Febră	adevărat	20	24
	fals	1480	1976
Durere de gât	adevărat	6	10
	fals	1494	1990
Dificultate în respirație	adevărat	1	1
	fals	1499	1999
Durere de cap	adevărat	6	11
	fals	1494	1989
Contact cu o persoană confirmată	adevărat	28	32
	fals	1472	1968

Figura 2.1: Explicitarea caracteristicilor setului de date

2.3 Preprocesarea datelor

Preprocesarea datelor este un proces fundamental în dezvoltarea modelului de învățare automată. Datele colectate sunt adesea interpretate în mod neclar cu valori în afara intervalului sau valori lipsă. Astfel de date pot induce în eroare rezultatul experimentului.

- **Gestionarea valorilor lipsă** [4] - În datele noastre, valorile lipsă au fost gestionate prin ștergerea rândurilor cu una sau mai multe coloane nule. Acest lucru conduce către un model robust, mai concret, evitând o posibilă strategie a mediei, care ar putea reduce erorile de bază și ar putea implicit să invalideze seturile de antrenament. În același timp, prin această metodă se pierde multă informație, constituind un dezavantaj. Se poate observa în figura 2.2 realizarea gestionării valorilor lipsă.



The figure displays two side-by-side terminal outputs. The left output shows the sum of null values for each column in a dataset, with most columns having 0 nulls, except for 'corona_result' (41218), 'age_60_and_above' (547644), 'gender' (92886), and 'test_indication' (24295). The right output shows the same dataset after removing rows with null values using 'dropna(axis=0)', where all columns now have 0 nulls.

Column	Sum of nulls (Left)	Sum of nulls (Right)
test_date	0	0
cough	0	0
fever	0	0
sore_throat	0	0
shortness_of_breath	0	0
head_ache	0	0
corona_result	41218	0
age_60_and_above	547644	0
gender	92886	0
test_indication	24295	0

Figura 2.2: Stânga: date cu valori nule, dreapta: date după eliminarea valorilor nule

2.4 Procesul de învățare automată

Potrivit lui Herbert Simon [5], învățarea este definită ca „orice schimbare a sistemului care îi permite să performeze mai bine a doua oară la repetarea aceleiași sarcini sau la o altă sarcină extrasă din aceeași populație”.

În contextul extragerii de date, cel mai important pas îl constituie abordarea, care în acest context se realizează cu ajutorul algoritmilor de învățare automată. Învățarea automată este un domeniu care se concentrează pe construcția de algoritmi care fac predicții bazate pe date, întrucât este preocupat să construiască programe cu scopul de a se îmbunătăți automat cu experiența.

În funcție de tipul de seturi de date care sunt utilizate, algoritmii de învățare automată sunt clasificați în două categorii principale: învățarea supervizată și învățarea nesupervizată.

Lucrarea stă la baza unei învățări supervizate, astfel că modelul este antrenat folosind un set de date de antrenament, unde învață despre fiecare tip de date. Următorul pas constă în a găsi cele mai optime valori ale hiper-parametrilor, în acest scop modelul fiind testat cu valori diferite ale hiper-parametrilor pe un set de date de validare, pentru a obține, în urma evaluării fiecăruia, cea mai bună performanță. Odată ce procesul de instruire este finalizat, modelul este testat pe baza datelor de testare și apoi prezice rezultatul [6]. Pentru implementarea unei clasificări, acest set se împarte în: set de antrenament, set de validare și set de testare.

Datele de antrenament sunt datele inițiale, ce au ca scop familiarizarea modelelor de învățare automată cu caracteristicile datelor introduse și sunt transmise algoritmilor de învățare automată cu scopul de a-i învăța cum să îndeplinească o sarcină dorită, să facă predicții.

Datele de validare sunt datele ce ajută în ajustarea hiper-parametrilor și alegerea modelului. Scopul acestora este de a facilita găsirea valorilor optime pentru hiper-parametrii modelului ales.

Datele de testare sunt datele ce prezintă o evaluare obiectivă a unui model final ce este compatibil pe setul de date de antrenament. Acesta are ca scop estimarea performanțelor algoritmului.

Capitolul 3

Tehnici abordate

3.1 Alegerea limbajului de programare

Pentru realizarea acestei teze am ales să folosesc Python, întrucât este un limbaj simplu, cu funcții și biblioteci extinse, ce vin în ajutorul învățării automate.

În această lucrare, au fost utilizate următoarele biblioteci Python:

1. **Pandas** - Oferă structuri de date expresive concepute să funcționeze atât cu date relaționale, cât și cu date etichetate. Este o bibliotecă care permite citirea și scrierea datelor între structurile de date [7]. Principalele utilizări ale acestei biblioteci au fost în citirea fișierului de date și prelucrarea acestuia.
2. **Numpy** - Este o bibliotecă pentru calcul științific. Numpy adaugă, de asemenea, capacități rapide de procesare a unei structuri de date matriciale [8].
3. **Sklearn** – Consider că este cea mai utilă bibliotecă pentru învățarea automată, întrucât ajută în aprofundarea algoritmilor ce stau la baza predicției, dispunând de diverși algoritmi pentru clasificare, grupare și regresie.

3.2 Metrici de performanță

Metricii de performanță sunt esențiali în măsurarea performanței unui model de învățare automată. Întrucât modelul stă la baza clasificării, am folosit acuratețea, sensibilitatea și specificitatea ca măsuri de performanță [9]. Pentru a calcula acești metrici am folosit următorii parametri:

- $TP= True\ Positive$, reprezentând rezultatele adevărat pozitive.
- $TN= True\ Negative$, reprezentând rezultatele adevărat negative.
- $FP= Fals\ Positive$, reprezentând rezultatele fals pozitive.
- $FN= Fals\ Negative$, reprezentând rezultatele fals negative.

3.2.1 Acuratețea

Este cea mai utilizată măsură de performanță pentru evaluarea tehnicilor de clasificare. Această măsură ne permite să înțelegem care model este cel mai bun la identificarea tiparelor în setul de antrenament pentru a oferi predicții mai bune în setul de date de testare.

$$\text{Acuratețea} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

Numărul de înregistrări ale pacienților	Mașina cu suport vectorial	Arborele de decizie	Cei mai apropiați K vecini
400	96.8 %	96.8 %	96.8 %
500	96.8 %	97 %	97 %
1000	97.1 %	97.8 %	97.3 %
1500	97.3 %	98 %	97.6 %
2000	98.1 %	98 %	97.8 %

Numărul de înregistrări ale pacienților	Random forest	Regresia logistică
400	97.3 %	96.8 %
500	97.5 %	96.8 %
1000	97.7 %	97.3 %
1500	97.8 %	97.5 %
2000	98.1 %	97.5 %

Figura 3.1: Acuratețea modelului

3.2.2 Sensibilitatea

Este măsură care descrie matematic acuratețea unui test, urmărind probabilitatea ca acesta să fie pozitiv, estimarea adevărat pozitivă. În cadrul lucrării, sensibilitatea este abilitatea de a clasifica corect un pacient fiind depistat „pozitiv” în cazul virusului Covid.

$$\text{Sensibilitatea} = \frac{TP}{TP + FN} \quad (3.2)$$

Numărul de înregistrări ale pacienților	Mașina cu suport vectorial	Arborele de decizie	Cei mai apropiați K vecini
400	0 %	0 %	0 %
500	0 %	6.25 %	6.25 %
1000	9.37 %	31.25 %	15.62 %
1500	18.7 %	40.62 %	28.12 %
2000	43.75 %	40.62 %	34.37 %

Numărul de înregistrări ale pacienților	Random forest	Regresia logistică
400	15.62 %	0 %
500	25 %	0 %
1000	31.25 %	15.62 %
1500	34.37 %	21.87 %
2000	43.75 %	21.87 %

Figura 3.2: Sensibilitatea modelului

3.2.3 Specificitatea

Este o măsură aflată în completarea sensibilității, existând un compromis între cele două. Astfel, specificitatea identifică acuratețea, ce studiază eventualitatea ca un test să fie negativ, evaluând parametrul adevărat negativ. În cadrul lucrării, specificitatea este abilitatea de a clasifica corect un pacient fiind depistat „negativ” în cazul virusului Covid.

$$\text{Specificitatea} = \frac{TN}{TN + FP} \quad (3.3)$$

Numărul de înregistrări ale pacienților	Mașina cu suport vectorial	Arborele de decizie	Cei mai apropiați K vecini
400	100 %	100 %	100 %
500	100 %	100 %	100 %
1000	100 %	100 %	100 %
1500	99.89 %	99.89 %	99.89 %
2000	99.89 %	99.89 %	99.89 %

Numărul de înregistrări ale pacienților	Random forest	Regresia logistică
400	100 %	100 %
500	99.89 %	100 %
1000	99.89 %	100 %
1500	99.89 %	100 %
2000	99.81 %	100 %

Figura 3.3: Specificitatea modelului

Rezultatele	Numărul de înregistrări ale pacienților	Mașina cu suport vectorial	Arborele de decizie	Cei mai apropiați K vecini
Adevărat pozitiv	400	0	0	0
	500	0	2	2
	1000	3	10	5
	1500	6	13	9
	2000	14	13	11
Adevărat negativ	400	968	968	968
	500	968	968	968
	1000	968	968	968
	1500	967	967	967
	2000	967	967	967
Fals pozitiv	400	0	0	0
	500	0	0	0
	1000	0	0	0
	1500	1	1	1
	2000	1	1	1
Fals negativ	400	32	32	32
	500	32	30	30
	1000	29	22	27
	1500	26	19	13
	2000	18	19	21

Rezultatele	Numărul de înregistrări ale pacienților	Random forest	Regresia logistică
Adevărat pozitiv	400	5	0
	500	8	0
	1000	10	5
	1500	11	7
	2000	14	
Adevărat negativ	400	968	968
	500	967	968
	1000	967	968
	1500	967	968
	2000	967	968
Fals pozitiv	400	0	0
	500	1	0
	1000	1	0
	1500	1	0
	2000	1	0
Fals negativ	400	27	32
	500	24	32
	1000	22	27
	1500	21	25
	2000	18	25

Figura 3.4: Numărul parametrilor în seturile de date stabilite

3.3 Implementare

Experimentul s-a desfășurat urmărind următoarele etape:

- Alegerea setului de date și preprocesarea datelor acestuia.
- Împărțirea datelor în seturi de antrenament, care conțin 400, 500, 1000, 1500, respectiv 2000 de înregistrări. Am ales să folosesc mai multe seturi de antrenament pentru a observa cum evoluează precizia modelelor (față de aceleași date de testare) pe măsură ce primesc mai multe date. Limita de 400 de înregistrări reprezintă valoarea minimă pe care modelele se pot antrena suficient, iar cea de 2000 reprezintă o barieră pe care am ales-o pentru a vedea rezultatul când numărul datelor de antrenare este de două ori mai mare decât numărul datelor de testare.
- Alegerea unui set de validare, pentru găsirea valorilor optime ale hiper-parametrilor. A fost selectat un set de 1000 de înregistrări.
- Alegerea unui set de testare. Am ales ca acesta să conțină 1000 înregistrări. În cele din urmă, se urmărește performanța fiecărui algoritm la fiecare set de antrenament, iar ulterior aceasta este comparată și evaluată pentru selectarea algoritmului optim.
- Urmărirea și analizarea celor mai importante caracteristici.

3.4 Configurarea algoritmilor

- Mașina cu suport vectorial

```
clf = svm.SVC(kernel = 'rbf', C = 1000, gamma = 1000)
```

Parametrii [10]:

kernel: va specifica ce tip de nucleu va fi utilizat în algoritm. Pentru că avem un set de date liniar, am ales să folosim 'rbf', funcția de bază radială.

C: este un parametru de regularizare, care indică o marjă în clasificarea greșită a setului de antrenament. Este un grad de predicție corectă, astfel că alegând un număr cât mai mare (1000) există șanse cât mai mici ca algoritmul să poată clasifica incorect.

gamma: indică influența pe care o are un exemplu dintr-un set de antrenament, urmărind distanța acestuia. Am ales o valoare cât mai mare (1000), pentru că folosind kernelul 'rbf' parametrul de regularizare și gamma trebuie să fie optimizați simultan.

- Arborele de decizie

```
clf = DecisionTreeClassifier()
```

- Cei mai apropiați k vecini

```
clf = KNeighborsClassifier(n_neighbors = 5, weights='distance',
algorithm='auto')
```

Parametrii [11]:

n_neighbors: reprezintă numărul de vecini care ajută în indicarea clasei punctului țintă. Am ales să folosesc un număr impar pentru a evita o posibilă egalitate.

weights: reprezintă ponderea folosită în predicții. Am ales „distance” pentru ca abordarea să fie în funcție de distanță (vecinii mai apropiați ai unui punct de interogare vor avea mai mult impact decât cei care sunt mai îndepărtați).

algorithm: indică indexarea structurii de date. Valoarea „auto” lasă algoritmul să decidă alegerea optimă privind căutarea vecinilor.

- Random Forest

```
clf = RandomForestClassifier(n_estimators=10, random_state=0)
```

Parametrii [12]:

n_estimators: reprezintă numărul de arbori ai pădurii aleatoare. Un număr cât mai mare oferă o performanță mai bună, dar codul este mai lent, astfel că am considerat optimă valoarea 10.

random_state: oferă siguranță în privința rezultatelor obținute, indicând că acestea pot fi reproduse.

- Regresia logistică

```
clf = LogisticRegression(max_iter=10000)
```

Parametrii [13]:

max_iter: indică numărul de iterații maxim care este necesar pentru a converge, astfel că am ales un număr cât mai mare.

3.5 Evaluarea modelelor

Evaluarea unui model depinde de capacitatea de predicție a acestuia, în această lucrare urmărindu-se performanța. Pentru seturile de date selectate am ales să folosesc: Mașina cu suport vectorial, Arborele de decizie, Cei mai apropiați K vecini (K-Nearest Neighbors), Random Forest, Regresia logistică, întrucât fiecare algoritm are un mod diferit de funcționare, avantaje, respectiv provocări pe care le aduce, evidențiate în capitolul următor.

Numărul de înregistrări ale pacienților	Mașina cu suport vectorial	Arborele de decizie	Cei mai apropiați K vecini
400	96.8 %	96.8 %	96.8 %
500	96.8 %	97 %	97 %
1000	97.1 %	97.8 %	97.3 %
1500	97.3 %	98 %	97.6 %
2000	98.1 %	98 %	97.8 %

Numărul de înregistrări ale pacienților	Random forest	Regresia logistică
400	97.3 %	96.8 %
500	97.5 %	96.8 %
1000	97.7 %	97.3 %
1500	97.8 %	97.5 %
2000	98.1 %	97.5 %

Figura 3.5: Comparație folosind măsurarea performanță - acuratețe

Privind acuratețea în urma verificării pe setul de testare, după ce modelele au fost antrenate pe diferite seturi (Figura 3.5), se pot identifica următoarele observații: cele mai bune rezultate se împart între 98.1 % (Mașina cu suport vectorial și Random forest), 98 % (Arborele de decizie), 97.8 % (Cei mai apropiați K vecini) și 97.5 % (Regresia logistică). Aceasta arată o diferență de 0.6 % legat de cea mai bună performanță între metodele de predicție (cea mai înaltă și cea mai scăzută), această diferență fiind destul de semnificativă.

De asemenea, comparând performanța în cazul numărului de înregistrări al unui model, se poate observa în cazul Mașinii cu suport vectorial o creștere de 1.3 % (între 400 și 2000 înregistrări), pe când Random forest are doar 0.3 %, ce ne indică faptul că Random forest are o performanță ridicată și în cazul unui set mai restrâns de antrenament.

3.6 Rezultate comparative

Curba caracteristicii de operare a receptorului (ROC) este o metodă populară de comparare a modelelor pentru diagnosticarea capacității unui sistem de clasificare binar,

deoarece pragul său de discriminare este variat [14].

Această curbă reprezintă o abordare grafică pentru a expune balansul dintre rata pozitiv adevărată (sensibilitatea) și cea pozitiv falsă (specificitatea) a clasificatorului [9].

Există mai multe puncte critice de-a lungul unei curbe ROC care pot fi interpretate astfel [9]:

- sensibilitate = 0, specificitate = 0: Modelul prezice fiecare instanță ca fiind o clasă negativă.
- sensibilitate = 1, specificitate = 1: Modelul prezice fiecare instanță a fi o clasă pozitivă.
- sensibilitate = 1, specificitate = 0: Modelul ideal.

Am ales să folosesc diferite seturi de testare, pentru o mai bună vizualizare a performanței, folosind Regresia logistică.

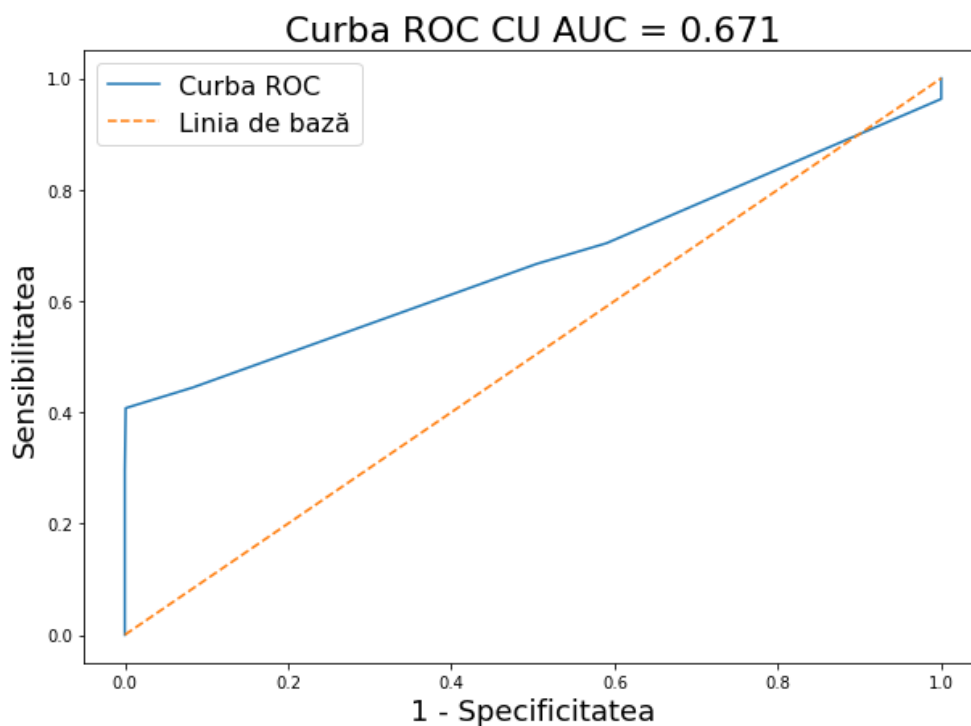


Figura 3.6: Curba ROC realizată pentru 400 înregistrări

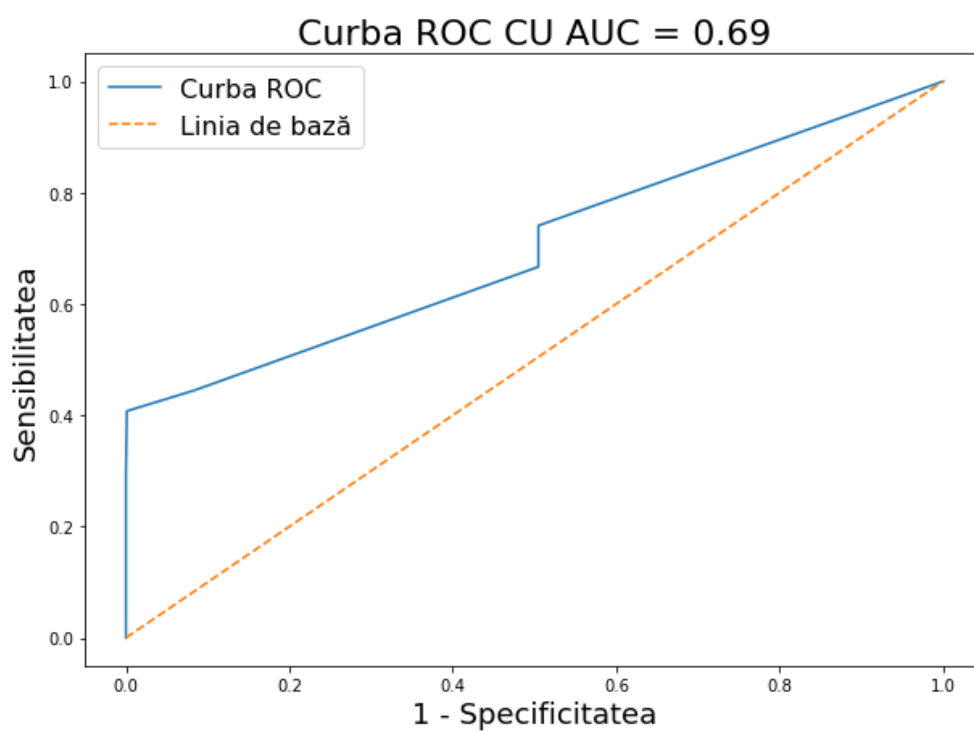


Figura 3.7: Curba ROC realizată pentru 500 înregistrări

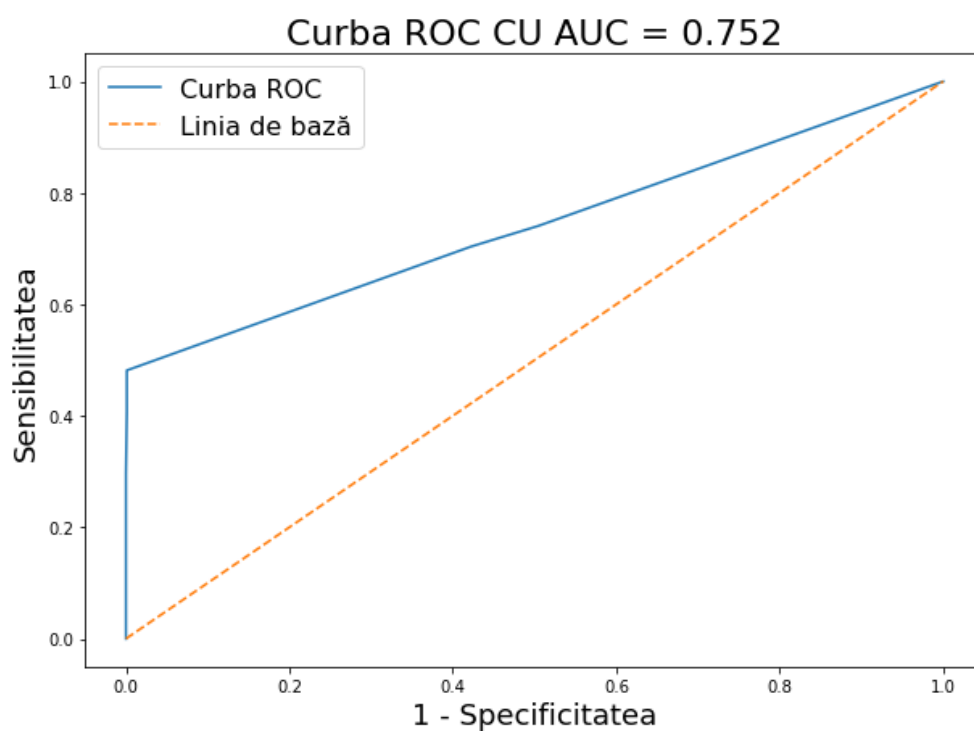


Figura 3.8: Curba ROC realizată pentru 1000 înregistrări

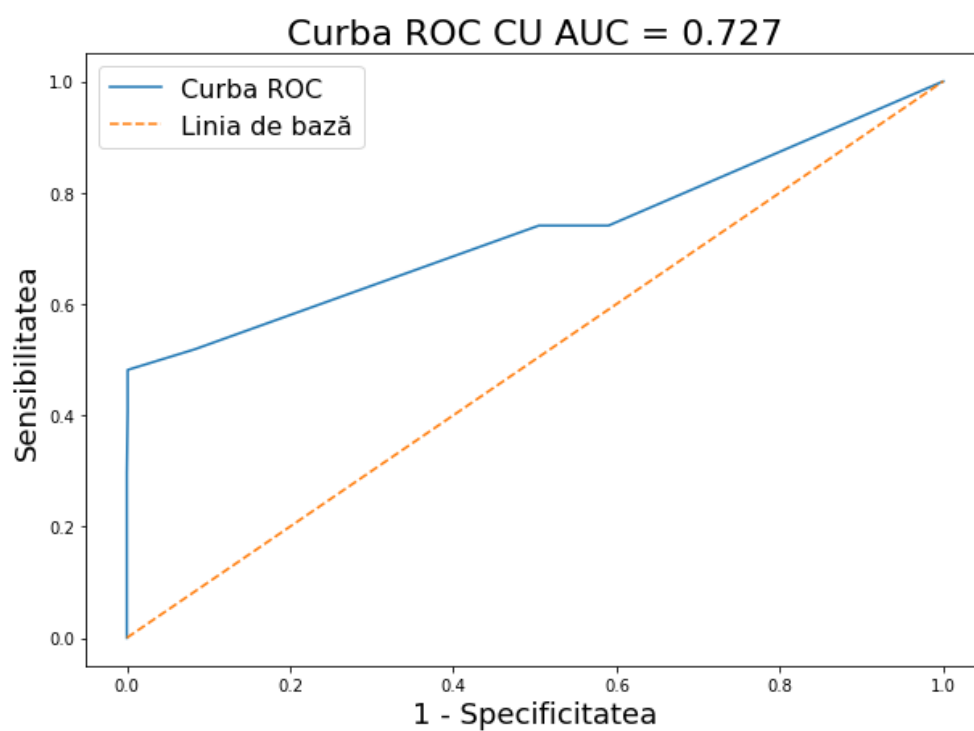


Figura 3.9: Curba ROC realizată pentru 1500 înregistrări

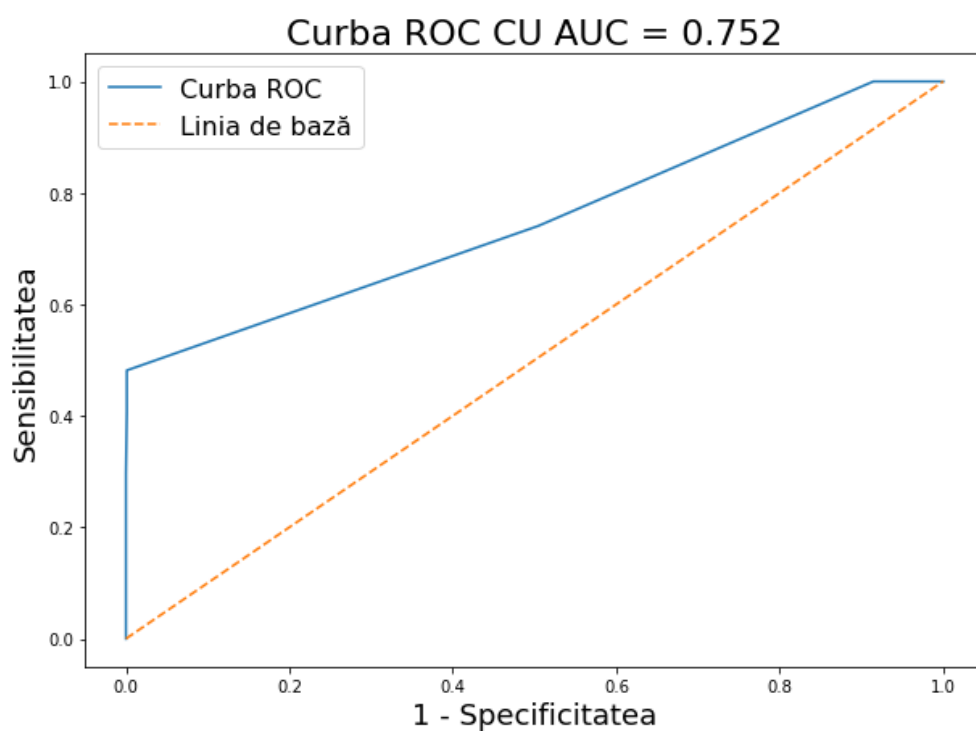


Figura 3.10: Curba ROC realizată pentru 2000 înregistrări

Analizând graficele, putem observa o performanță bună a clasificării, pentru toate înregistrările, întrucât modelul de clasificare este situat cât mai aproape de colțul din stânga al diagramei (în cazul unui model care ar face presupuneri aleatorii, ar trebui să fie de-a lungul diagonalei principale, conectând punctele **sensibilitate = 0, specificitate = 0** și **sensibilitate = 1, specificitate = 1**).

Aria de sub curba ROC (AUC) oferă o altă abordare pentru evaluare. În acest caz, cu cât această suprafață se apropie mai mult de valoarea 1, cu atât modelul este mai bun. Dacă ajunge la 1, putem considera că este perfect.

Urmărind graficele, putem observa valori ce tind să fie din ce în ce mai bune odată cu creșterea numărului de înregistrări (0.671, 0.69, 0.752, 0.727, 0.752). În cazul în care vrem să antrenăm un model sub 400 înregistrări, aria de sub curba ROC poate atinge valoarea maximă aproximativ 0.5, ce ne induce că modelul face presupuneri aleatorii, nu este antrenat suficient. Acest lucru se poate observa și în figura 3.11, unde pentru 350 înregistrări avem valoarea 0.49.

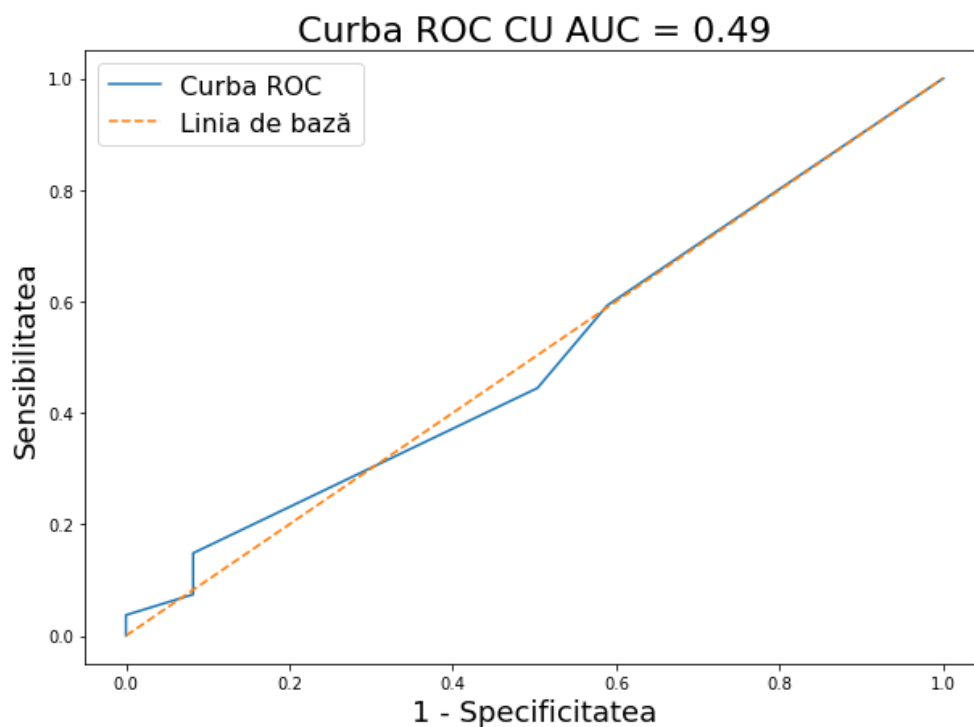


Figura 3.11: Curba ROC realizată pentru 350 înregistrări

Conform curbei ROC, zona de sub curbă (AUC) are trei interpretări [15]:

- sensibilitatea medie pentru toate posibilele rezultate false pozitive.
- specificitatea medie pentru toate posibilele rezultate pozitive adevărate.
- probabilitatea ca o variabilă predictor să poată fi utilizată pentru a selecta corect condiția rezultatului dintr-o pereche selectată aleatoriu care conține un pacient cu condiția rezultatului și unul fără.

3.7 Determinarea celor mai importante caracteristici

Pentru a clasifica cele mai importante caracteristici ale modelului am folosit diagrama SHAP, o abordare teoretică pentru a explica rezultatul oricărui model de învățare automată [16]. Prin această diagramă:

- Caracteristicile sunt organizate după valorile lor medii absolute.
- Fiecare punct corespunde unei persoane individuale din studiu.
- Poziția fiecărui punct pe axa X arată impactul pe care îl are această caracteristică asupra predicției pentru un anumit individ.

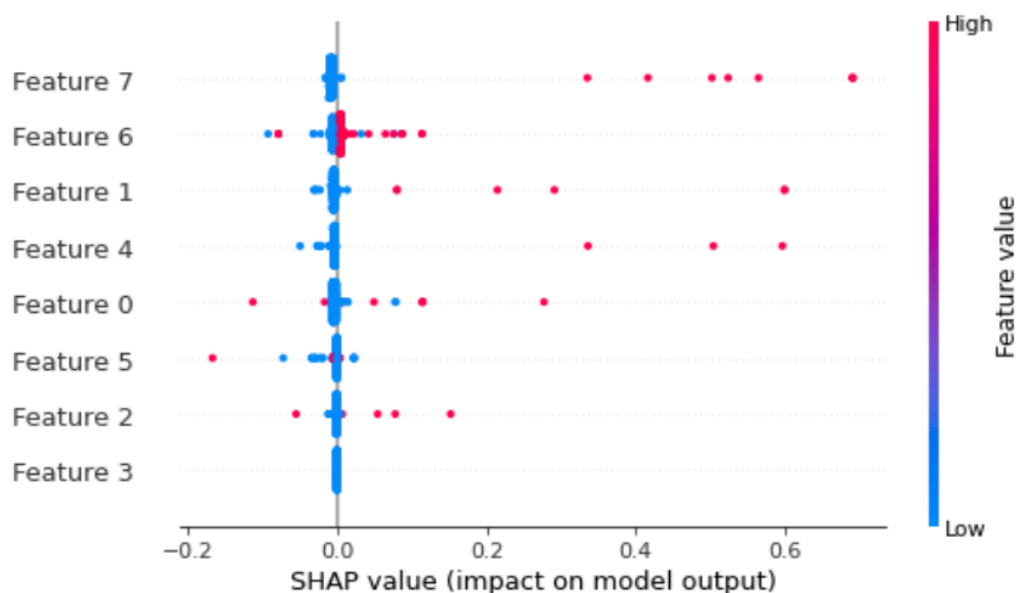


Figura 3.12: Diagrama SHAP

Simptomele din figura 3.12 au fost numerotate de la 0 la 7 după cum urmează: tuse, febră, durere de gât, dificultate în respirație, durere de cap, vârsta ≥ 60 ani, sexul, contact cu o persoană confirmată.

Diagrama a fost realizată pe un test de 2000 pacienți și se poate observa că febra și tusea au fost cheia pentru a prezice prezența bolii. După cum era de așteptat, contactul strâns cu o persoană confirmată că are COVID-19 a fost, de asemenea, o caracteristică importantă, coroborând astfel transmisibilitatea ridicată a virusului și evidențiind importanța distanțării sociale.

3.8 Configurarea unei interfețe grafice

Am ales să implementez o interfață grafică, pentru a evidenția cu ușurință rularea algoritmilor în vederea evaluării modelelor.

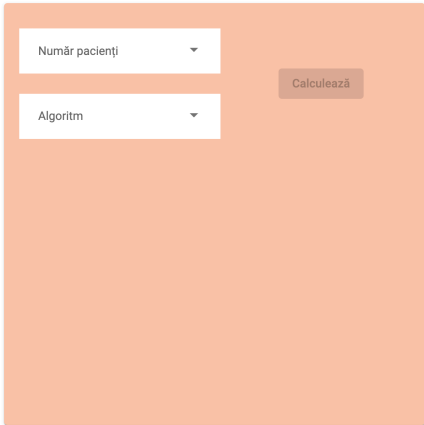
3.8.1 Tehologii folosite

- **Angular** [17]: Este un framework dedicat creării aplicațiilor web de tipul SPA (single-page application). Interfața folosește ca parte de front-end acest framework, iar legătura cu partea de back-end se realizează prin existența unui serviciu. Acesta este injectat în componentă și face două request-uri de tip GET cu scopul de a returna precizia, sensibilitatea și specificitatea și respectiv plot-ul.
- **Python**: Pentru partea de back-end am folosit limbajul de programare Python, după cum am menționat în secțiunea 3.1.

3.8.2 Modul de funcționare

- Accesând aplicația, va trebui să selectăm numărul de pacienți (setul de antrenament) și algoritmul.

IDENTIFICAREA PACIENȚILOR DIAGNOSTICAȚI CU COVID-19



The image shows a web application interface with a light orange background. At the top, there is a header with the text "IDENTIFICAREA PACIENȚILOR DIAGNOSTICAȚI CU COVID-19". Below the header, there is a form with two dropdown menus. The first dropdown menu is labeled "Număr pacienți" and the second is labeled "Algoritm". To the right of these dropdowns is a button labeled "Calculează".

Figura 3.13: Selectarea numărului de pacienți și a algoritmului

- Valorile alese vor fi trimise pe serverul de Python, unde se vor calcula precizia, sensibilitatea si specificitatea.

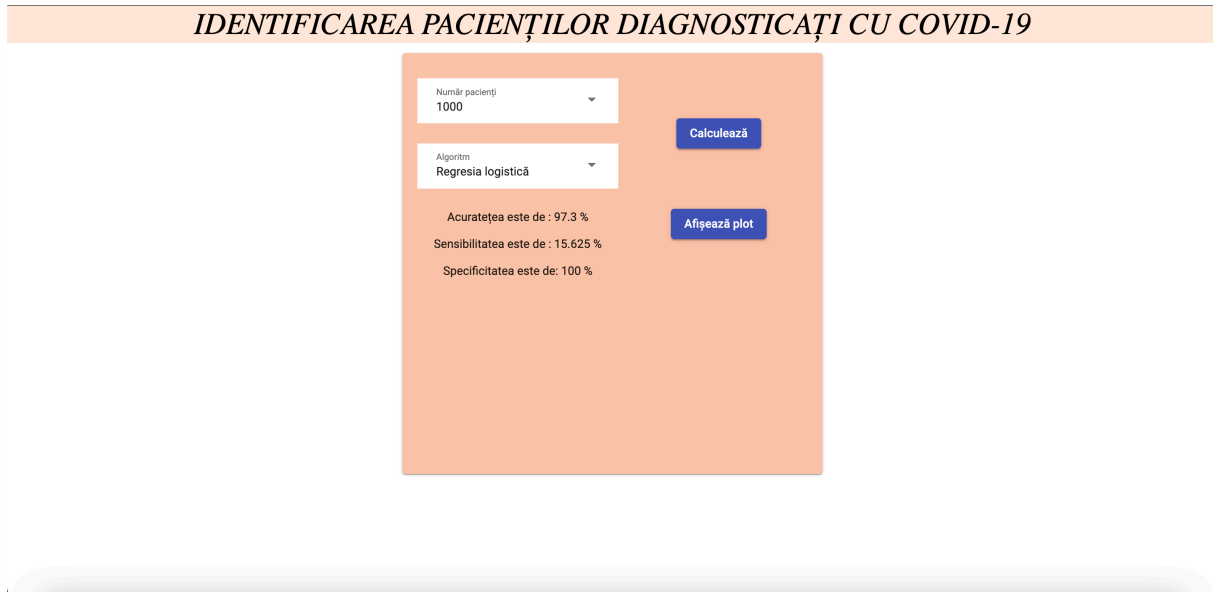


Figura 3.14: Afișarea preciziei, a sensibilității și specificității

- Ca ultim pas, putem genera și Curba caracteristicii de operare a receptorului (Curba ROC).

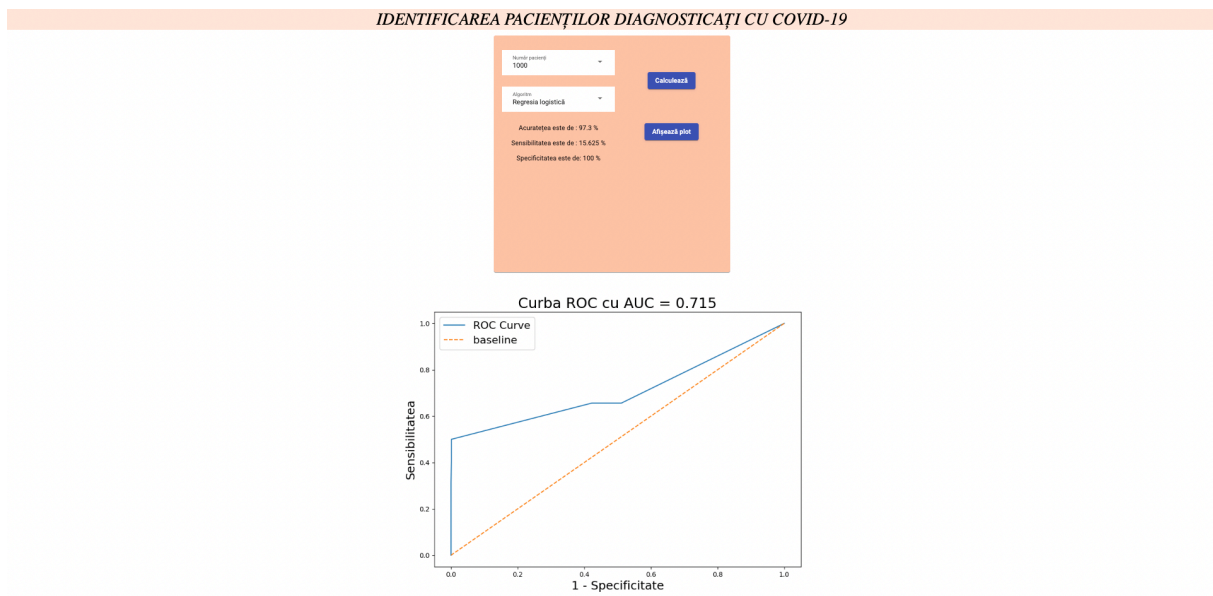


Figura 3.15: Generarea curbei ROC

Capitolul 4

Algoritmi utilizați

4.1 Mașina cu suport vectorial

Mașina cu suport vectorial realizează clasificarea prin construirea unui hiperplan în spațiu multidimensional pentru a separa diferite clase.

Scopul final al modelării mașinii cu suport vectorial este de a genera hiperplanul optim într-o manieră iterativă, care este utilizat pentru a minimiza o eroare și de a selecta un hiperplan cu marginea maximă posibilă între vectorii suport din setul de date. Ideea de bază este de a găsi un hiperplan marginal maxim care împarte în clase în cel mai bun mod setul de date. Vectorii care sunt în apropierea hiperplanului sunt vectorii suport.

„Vectori suport”: Vectorii suport sunt punctele de date, care sunt cele mai apropiate de hiperplan. Aceste puncte vor defini mai bine linia de separare prin calcularea marjelor. Aceste puncte sunt mai relevante pentru construcția clasificatorului [18].

„Hiperplan”: Un hiperplan este un plan de decizie care separă între un set de obiecte cu apartenență la clasă diferită [18].

„Marja”: O marjă este un decalaj între cele două linii de pe cele mai apropiate puncte ale clasei. Aceasta este calculată ca distanța perpendiculară de la linie la vectorii de sprijin sau punctele cele mai apropiate. Dacă marja este mai mare între clase, atunci aceasta este considerată o marjă bună, în timp ce o marjă mai mică este considerată rea [18].

4.1.1 Avantaje

- Algoritmul se bazează pe o teorie stabilă și nu ține cont de mărimea setului de date, având o precizie optimă și în cazul unui set de mici dimensiuni. În cazul setului de date introdus, se poate observa o acuratețe de 96.8 % pentru 400 pacienți.
- Folosește vectori suport ca și puncte de antrenament în cazul predicției, fiind eficient ca și memorie.

4.1.2 Provocări

- Mașina cu suport vectorial necesită un timp îndelungat de antrenament în cazul seturilor mari de date. În acest sens, am ales seturi mici de date, până la 2000 înregistrări.

Tabelul 4.2 reprezintă acuratețea pentru fiecare set de înregistrări realizată de algoritmul Mașina cu suport vectorial (SVM).

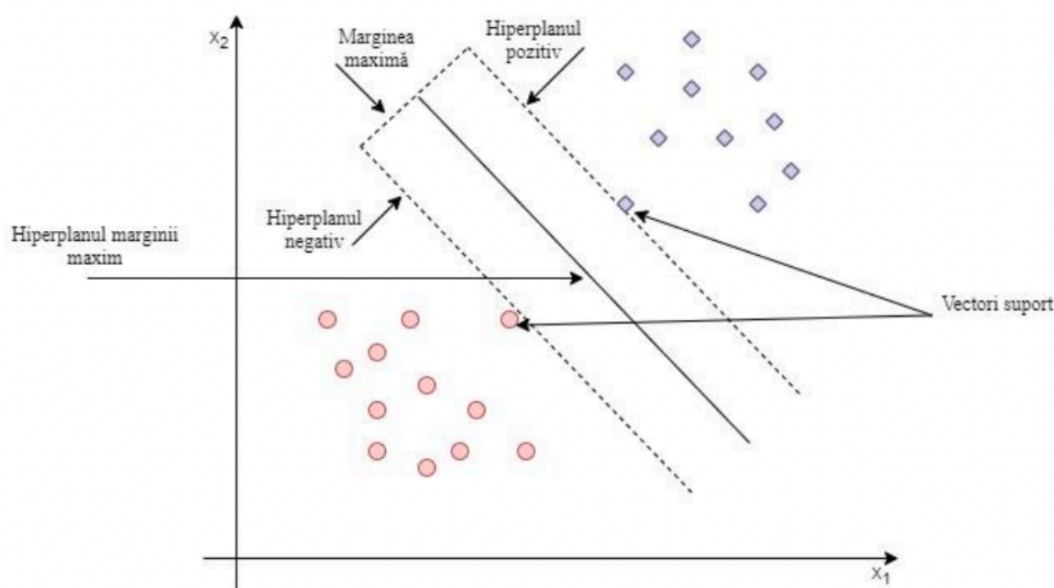


Figura 4.1: Mașina cu suport vectorial: Găsirea hiperplanului optim

Numărul de înregistrări ale pacienților	Acuratețea
400	96.8 %
500	96.8 %
1000	97.1 %
1500	97.3 %
2000	98.1 %

Figura 4.2: Rezultate de acuratețe ale mașinii cu suport vectorial (SVM)

4.2 Arborele de decizie

Arborele de decizie se bazează pe următoarea idee: efectuarea predicțiilor folosind o succesiune de decizii simple. Un model de arbore de decizie constă într-un ansamblu de decizii (binare) organizate în mod ierarhic, ordonat. Prin urmare, un arbore de decizie poate fi definit formal ca un graf aciclic direcționat, compus dintr-un set de muchii direcționate și unul de noduri.

Orice nod codifică o decizie (binară) și este conectat printr-o muchie direcționată la cel mult un nod părinte de la nivelul superior și cel puțin două noduri copii de la nivelul inferior.

„**Dirijat**” implică faptul că:

- arborele poate fi traversat doar folosind o cale descendentă, adică direcția părinte-copii
- nodurile de la diferite niveluri nu sunt intervertibile, în sensul că nu își pot schimba ordinea.

„**Aciclic**” înseamnă că nu există cicluri într-un model de arbore. În timp ce nodul din vârful unui copac se numește rădăcină, nodurile din partea de jos sunt numite frunze. Se poate traversa arborele în jos, urmând un drum unic care este determinat de deciziile luate la fiecare nod traversat, până când ajunge la o frunză, așa cum este ilustrat în figura 4.3. În timpul fazei de învățare, datele care au ajuns la o anumită frunză sunt folosite pentru a modela distribuția posterioară „local”. În timpul fazei de testare, aceste distribuții posterioare permit să se facă predicții despre noile observații nevăzute care ajung la o anumită frunză.

4.2.1 Avantaje

- Interpretarea este una accesibilă, algoritmul fiind ușor de înțeles. Acest lucru este susținut și de Tjen-Siem Lim, care a efectuat un studiu comparativ al arborilor de decizie față de alți algoritmi de învățare automată și a concluzionat că algoritmul posedă o combinație foarte bună privind viteza de clasificare și rata de eroare [19].

4.2.2 Provocări

- Nu este eficient și potrivit pentru seturile mari de date. În cazul unui set de dimensiune mare un singur arbore devine complex și poate duce la supraadaptare. În acest caz este indicat să fie folosit Random Forest, ce conține mai mulți arbori de decizie.
- Este un algoritm instabil. În cazul adăugării unui nou punct de date este posibilă regenerarea arborelui general, astfel că toate nodurile trebuie recreate și recalulate.

Tabelul 4.4 reprezintă acuratețea pentru fiecare set de înregistrări realizată de algoritmul Arborelui de decizie.

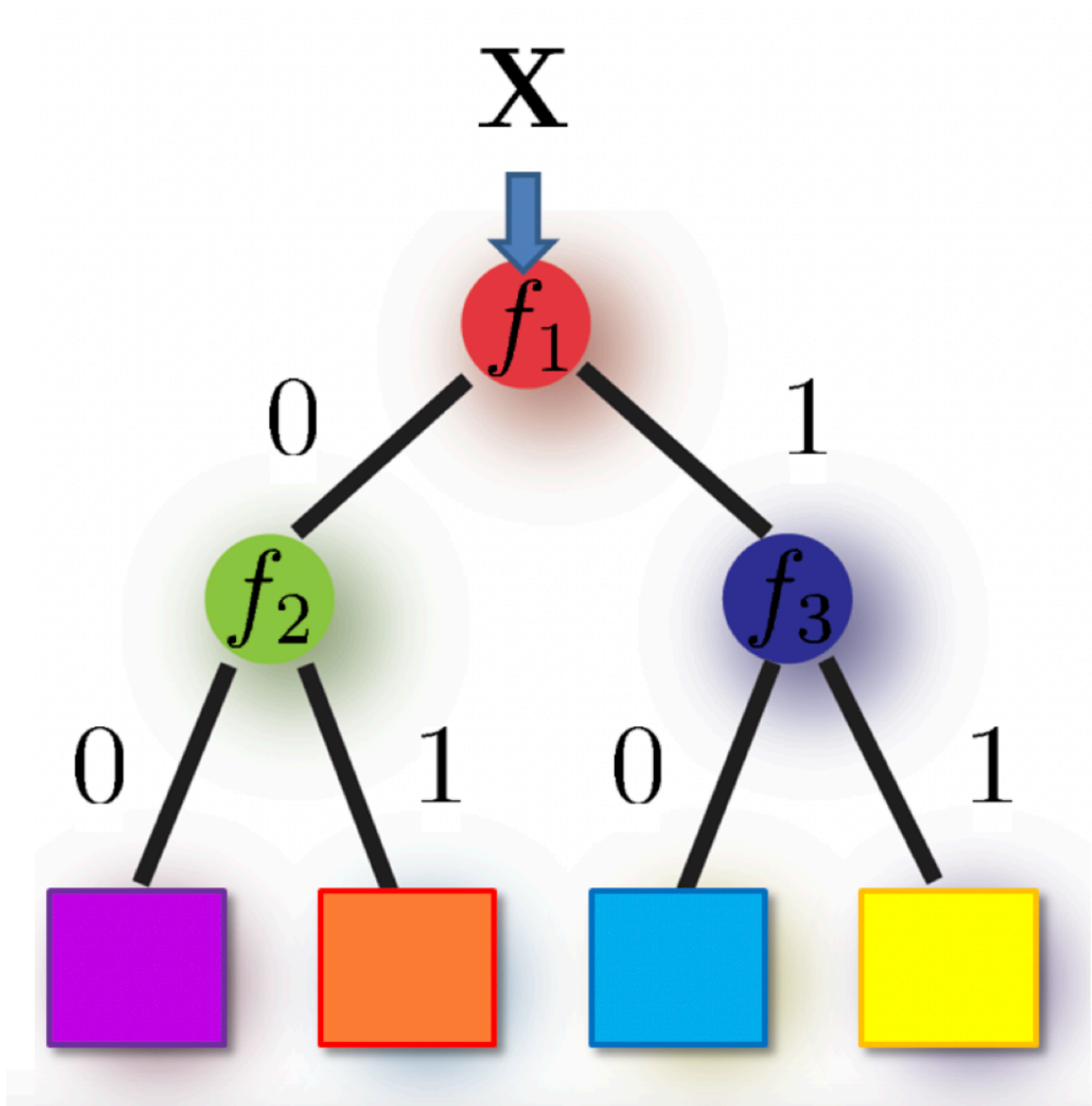


Figura 4.3: Arborele de decizie

Numărul de înregistrări ale pacienților	Acuratețea
400	96.8 %
500	97 %
1000	97.8 %
1500	98 %
2000	98 %

Figura 4.4: Rezultate de acuratețe ale arborelui de decizie

4.3 Cei mai apropiați K vecini (K-nearest neighbors)

Este un algoritm non-parametric, „bazat pe instanță”, ceea ce înseamnă că trebuie să-și păstreze fiecare observație de antrenament. Acesta prezice noi observații de eșantion prin căutarea celor mai asemănătoare eșantioane de antrenament.

Algoritmul KNN folosește ca principiu „asemănarea caracteristicilor” cu scopul de a prezice valorile a oricăror puncte noi de date. Prin acest lucru înțelegem că unui nou punct i se alocă o valoare depinzând de cât de mult corespunde cu punctele din setul de antrenament.

Principiul de funcționare se poate urmări mai concret, având două categorii și un punct de date nou (figura 4.5). Pentru a atribui o categorie acelui punct este necesar să alegem un număr K ai vecinilor și să luăm K vecini cei mai apropiați conform distanței euclidiene (am ales $K=5$, figura 4.6). Categoria din care punctul de date face parte este cea pentru care numărul vecinilor este maxim (figura 4.7).

Prin calcularea distanței euclidiene am obținut cei mai apropiați vecin ca fiind cel din categoria A.

4.3.1 Avantaje

- Este o metodă neparametrică, însemnând că distribuția parametrilor nu trebuie să fie cunoscută [20].
- Este o abordare non-parametrică. Nu este necesară adaptarea/formarea modelului.
- Este stabil, micile modificări ale datelor de antrenament nu conduc la diferite rezultate de clasificare [20].
- Algoritmul este simplu și ușor de implementat, deoarece necesită doar doi parametri: valoarea K și funcția de distanță. Acest lucru poate fi observat și în cadrul Secțiunii 3.4.

4.3.2 Provocări

- Algoritmul nu funcționează bine în cazul atributelor din seturile de antrenament ce au dimensiuni mari, deoarece distanța trebuie calculată în fiecare punct de fiecare dată când algoritmul întâlnește un nou punct de date.
- Dacă setul nostru de date necesită un K care este un număr mare, va crește costul de calcul al algoritmului și implicit complexitatea. Din acest motiv am considerat potrivită valoarea 5 pentru K .
- Nu poate gestiona datele dezechilibrate. Când datele sunt dezechilibrate (există mult mai multe date aparținând unei anumite categorii decât restul categoriilor),

algoritmul va fi părtinitor. Prin urmare, trebuie tratat în mod explicit. Pentru a obține un algoritm cât mai curat și precis am căutat un echilibru în cazul caracteristicilor seturilor de antrenament, care poate fi observat în tabelul 2.1.

- Cei mai apropiați K vecini presupune că punctele de date similare sunt aproape unele de celelalte. Prin urmare, modelul este susceptibil la valori aberante. Câteva valori aberante dintr-o anumită categorie pot atrage noile date către ea chiar și în cazurile în care noile date aparțin unei alte categorii. Astfel, am ales un set de date binar, pentru a avea un rezultat cât mai concludent, mai exact.

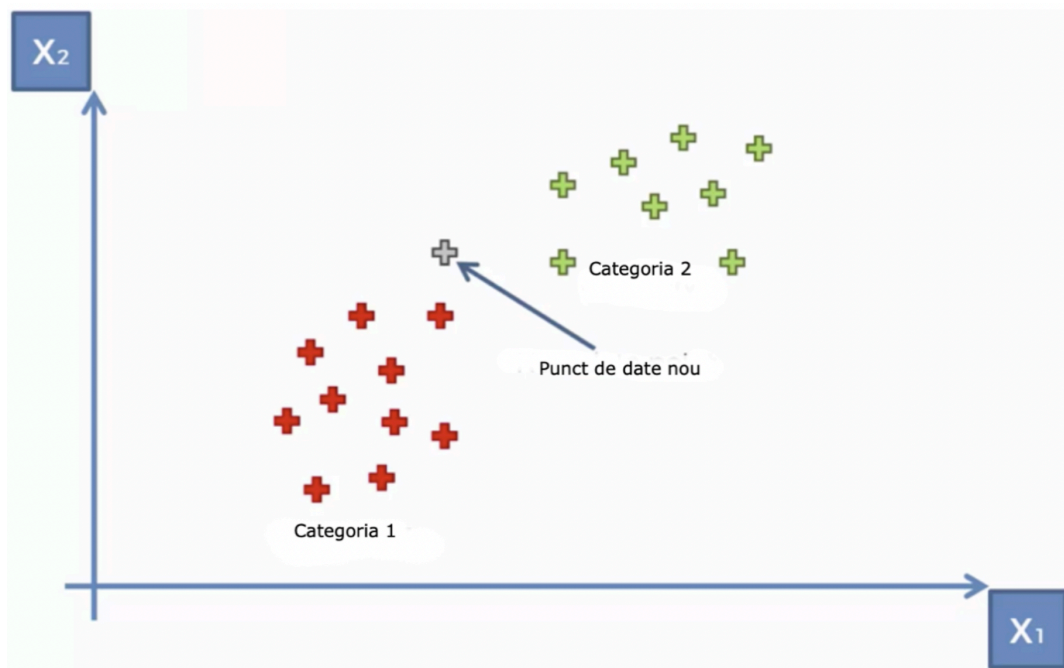


Figura 4.5: Cei mai apropiați K vecini

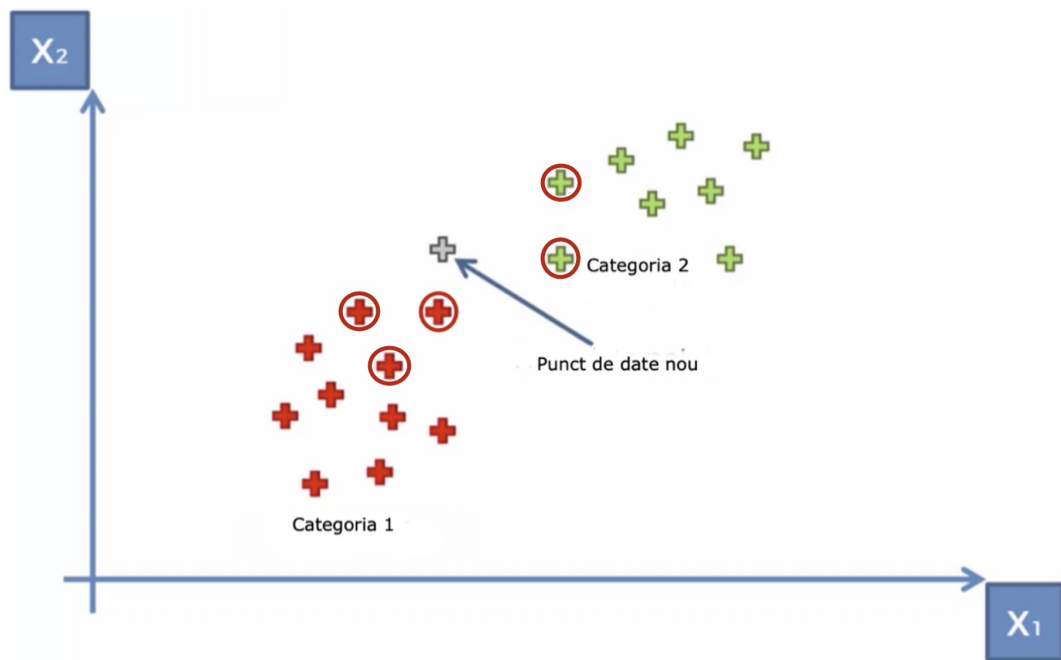


Figura 4.6: Cei mai apropiați K vecini

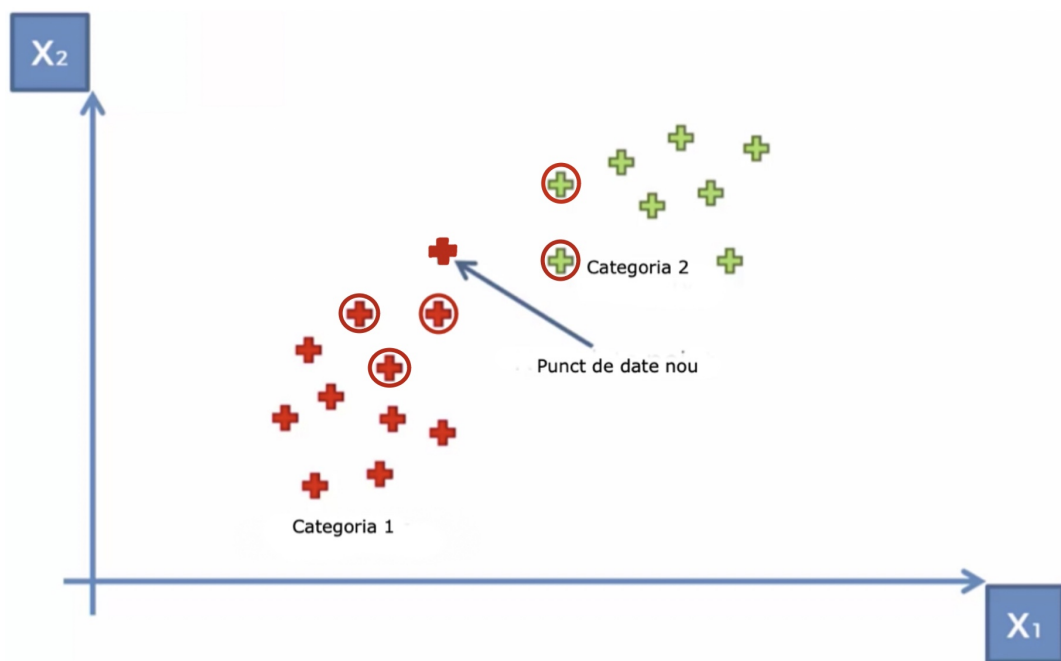


Figura 4.7: Cei mai apropiați K vecini

Numărul de înregistrări ale pacienților	Acuratețea
400	96.8 %
500	97 %
1000	97.3 %
1500	97.6 %
2000	97.8 %

Figura 4.8: Rezultate de acuratețe ale celor mai apropiați K vecini (KNN)

4.4 Random Forest

Este un algoritm care construiește și îmbină mai mulți arbori de decizie, cu scopul de a obține o predicție mai sigură, exactă și mai solidă, stabilă.

Termenul „*Forest*” provine din faptul că deține „o pădure” de arbori de decizie. Datele din aceștia sunt unite, astfel că se realizează o predicție mai precisă. Acest lucru este demonstrat prin faptul că în cazul unui arbore de decizie individual, acesta are un singur rezultat și o gamă restrânsă de grupuri, pe când pădurea, printr-un număr mai mare de grupuri și decizii, asigură un rezultat mai precis.

În ciuda principiului asemănător de funcționare, acești algoritmi se manifestă diferit, astfel că un arbore de decizie tinde să creeze reguli, pe care le folosește pentru a lua decizii, pe când o pădure aleatoare va alege aleatoriu caracteristici și va face observații, va construi o pădure de arbori de decizie și apoi va face o medie a rezultatelor.

4.4.1 Avantaje

- Este un algoritm versatil, care poate fi folosit atât în cazul regresiei, cât și a clasificării.
- Este un algoritm precis, întrucât utilizează un număr de arbori cu diferențe semnificative între subgrupuri.

4.4.2 Provocări

- Rezultatele au un timp mare de așteptare. Prin faptul că algoritmul construiește mulți arbori, automat crește și acuratețea predicțiilor. Cu toate acestea, încetinește viteza procesului, deoarece construiește sute sau mii de arbori, astfel că îl face ineficient, în special pentru predicții în timp real.

- Sunt greu de interpretat. Nu sunt explicabile, așa că este greu de înțeles cum sau de ce au ajuns la o anumită decizie. Această impenetrabilitate înseamnă că modelul trebuie pur și simplu să fie de încredere, precum și rezultatele sunt acceptate așa cum sunt.

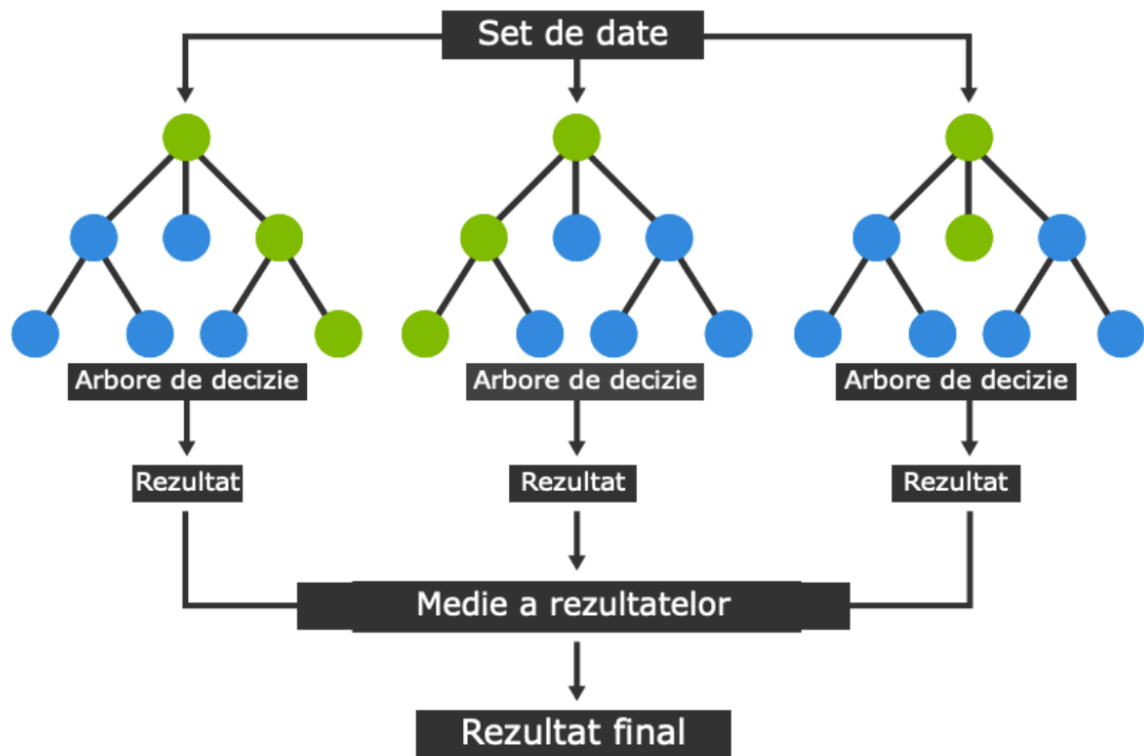


Figura 4.9: Random Forest

Numărul de înregistrări ale pacienților	Acuratețea
400	97.3 %
500	97.5 %
1000	97.7 %
1500	97.8 %
2000	98.1 %

Figura 4.10: Rezultate de acuratețe ale Random Forest

4.5 Regresia logistică

Este un algoritm folosit pentru a caracteriza datele și care urmărește rezolvarea problemelor de clasificare binară, fiind o metodă de analiză statică. Folosind regresia logistică, putem estima relațiile între unul sau mai mulți parametri independenți (predictori) și o variabilă binară dependentă (predicția). Este o extensie a modelului de regresie liniară în cazul problemelor de clasificare.

Este concepută după patru ipoteze principale [21]:

- Variabila țintă este categorică.
- Termenii de eroare sunt independenți.
- Predictorii sunt necorelați.
- Variabilele sunt relevante.

4.5.1 Avantaje

- Este un algoritm simplu. Regresia logistică poate fi văzută drept unul dintre cei mai ușori algoritmi, din faptul că este puțin mai implicată decât regresia liniară, care este unul dintre cei mai simpli algoritmi predictivi.
- Este un algoritm transparent. Putem vedea procesul acestuia pas cu pas și înțelege ce se întâmplă, în comparație cu restul algoritmilor urmăriți în lucrare (Mașina cu suport vectorial, Arborele de decizie, Cei mai apropiați K vecini, Random Forest), care sunt mult mai greu de urmărit.

4.5.2 Provocări

- Se poate ajunge ușor la supraadaptarea modelului. Știm că regresia logistică este un model de analiză statistică care încearcă să prezică rezultate probabilistice precise pe baza unor caracteristici independente. Însă, pe seturile de date cu dimensiuni mari, acest lucru poate duce la supraadaptarea modelului (în cazul setului de antrenament), ceea ce înseamnă supraestimarea acurateții predicțiilor pe setul de antrenament. Prin urmare, sunt șanse mari ca modelul să nu poată prezice rezultate precise pe setul de testare. Acest lucru este evidențiat mai ales în cazul în care modelul este antrenat pe date de antrenament puține cu multe caracteristici.
- Oferă incertitudine când vine vorba despre importanța caracteristicilor. Importanța unei caracteristici reflectă cum și cât de mult interacționează aceasta cu răspunsul. Folosind regresia logistică nu suntem atât de siguri de acest lucru, întrucât importanța nu depinde doar de asocierea dintre o variabilă independentă și variabila dependentă, ci și de legătura cu alte variabile independente.

Figura 4.11 reprezintă acuratețea pentru fiecare set de înregistrări realizată de algoritmul Regresiei logistice.

Numărul de înregistrări ale pacienților	Acuratețea
400	96.8 %
500	96.8 %
1000	97.3 %
1500	97.5 %
2000	97.5 %

Figura 4.11: Rezultate de acuratețe ale Regresiei logistice

Capitolul 5

Concluzii

Predicția COVID-19 prin utilizarea Machine Learning ar putea ajuta la creșterea vitezei de identificare a bolii, ceea ce duce la reducerea ratei mortalității, întrucât, în multe cazuri, dacă virusul este descoperit din timp în organism, el poate fi tratat mai ușor, fără să se extindă în tot organismul.

Analizând rezultatele obținute în urma experimentelor, **Random Forest** a fost identificat cu performanțe mai bune în comparație cu alți algoritmi, având cea mai mare acuratețe pe datele de testare în privința tuturor înregistrărilor. Am putut observa diferențe semnificative în cazul unor modele cu cât setul de antrenament a fost mai mare, astfel că de la 400 înregistrări la 2000, **Mașina cu suport vectorial** a înregistrat o creștere de 1.3 %, **Arborele de decizie** 1.2 % și **Cei mai apropiați K vecini** 1 %, pe când **Regresia liniară** a avut doar 0.7 %, iar **Random Forest** 0.3%.

De asemenea, am observat cele mai importante caracteristici privind prezicerea și detectarea virusului, acestea fiind febra, tusea și respectiv contactul strâns cu o persoană confirmată că are COVID-19, pentru care consider că distanțarea socială ar fi un mare ajutor, pentru a opri extinderea.

Cu toate acestea, cu mai mult timp, efort, date și o mai bună înțelegere a funcționalităților mai complexe oferite de fiecare bibliotecă, nu ar exista nicio îndoială că precizia ar putea deveni și mai mare.

Dezvoltări ulterioare

În contextul menționat anterior, îmi doresc să diversific baza de date și să efectuez studii suplimentare, care împreună cu tehnicile prezentate în această lucrare pot avea ca urmare rafinarea clasificării, respectiv obținerea unei precizii cât mai bune și exacte. Un exemplu în acest caz poate fi deosebirea între COVID-19 și gripă, având unele simptome comune [22] (febră, tuse, durere în gât, oboseală, nas înfundat, dureri de cap), care pot crea confuzii.

Bibliografie

- [1] *Propunere de REGULAMENT AL PARLAMENTULUI EUROPEAN ȘI AL CONSILIULUI*, URL: http://www.cdep.ro/afaceri_europene/CE/2022/COM_2022_50_R0_ACT_part1_v2.pdf, accesat: 01.01.2022.
- [2] *Covid-19*, URL: <https://www.consilium.europa.eu/ro/policies/coronavirus/>, accesat: 04.01.2022.
- [3] *Database Israeli Ministry of Health*, URL: <https://data.gov.il/dataset/covid-19>, accesat: 10.10.2021.
- [4] Jason W Osborne, *Best practices in Data Cleaning*, SAGE Publications, 2013.
- [5] J.G. Carbonell T.M. Mitchell R.S. Michalski, *Machine learning, an artificial intelligence approach*, Tioga Publishing Co, cap. 10.
- [6] *Supervised Machine Learning*, URL: <https://www.javatpoint.com/supervised-machine-learning>, accesat: 25.12.2021.
- [7] Wes McKinney și PD Team, *Pandas—Powerful Python Data Analysis Toolkit*, 2015.
- [8] Wes McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.
- [9] VIPIN KUMAR PANG.N ING TAN MICHAEL STEINBACH, *Introduction to Data Mining*, Pearson Education, Inc., 2006.
- [10] *SVM Scikit*, URL: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>, accesat: 08.10.2021.
- [11] *KNN Scikit*, URL: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>, accesat: 08.10.2021.
- [12] *Random Forest Scikit*, URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>, accesat: 08.10.2021.
- [13] *Logistic Regression Scikit*, URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, accesat: 08.10.2021.
- [14] *Receiver Operating Characteristic*, URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic, accesat: 04.01.2022.

- [15] Obuchowski NA., „ROC analysis”, în (Feb. 2005), DOI: [10.2214/ajr.184.2.01840364](https://doi.org/10.2214/ajr.184.2.01840364).
- [16] *SHapley Additive exPlanations*, URL: <https://shap.readthedocs.io/en/latest/index.html>, accesat: 04.01.2022.
- [17] *Angular framework*, URL: <https://angular.io/docs>, accesat: 10.02.2022.
- [18] *Support Vector Machine*, URL: <https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python#svm>, accesat: 15.02.2022.
- [19] Murthy, *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, Data Mining and Knowledge Discovery*, 1998.
- [20] Breiman L, *Bagging predictors. Machine Learning*, 1996.
- [21] *Logistic Regression*, URL: <https://sites.google.com/site/kaust229machinelearning/>, accesat: 04.01.2022.
- [22] *COVID-19: Identificarea simptomelor*, URL: <https://www.health.gov.au/sites/default/files/documents/2020/08/coronavirus-covid-19-identificarea-simptomelor-identifying-the-symptoms-covid-19-identificarea-simptomelor.pdf>, accesat: 06.06.2022.