

# Sistemi di diagnostica e predizione

...

Eletti Davide - 698807 - d.eletti@studenti.uniba.it  
Calabrese Luciana - 704347 - l.calabrese21@studenti.uniba.it

[https://github.com/Luciana-99/icon\\_project.git](https://github.com/Luciana-99/icon_project.git)

# Sistema di diagnostica

Il **sistema del progetto** cercherà di diagnosticare la malattia del covid o dell'influenza, basandosi sui dati forniti dall'utente, il quale viene coinvolto nella risposta a una serie di domande mirate a individuare i sintomi di cui l'utente soffre e le possibili relazioni che questi hanno col virus, in modo da stabilire la probabilità che ha l'utente di essere affetto dal virus. Adoperiamo con il **forward chaining** andando a chiedere prima le domande, per poi determinare se è presente o meno una forma di malattia nella persona. L'idea generale si basa su una regola di derivazione, una forma generalizzata della regola di inferenza chiamata *modus ponens*:

Se " $h \leftarrow a_1 \wedge \dots \wedge a_m$ " è una clausola definita nella base di conoscenza e ogni  $a_i$  è stato derivato, allora  $h$  può essere derivato.

Dove:

- $h$  è la testa dell'atomo,
- $a_1 \wedge \dots \wedge a_m$  è il corpo della clausola, formato da  $a_i$  atomi

Se  $m > 0$ , la clausola è detta regola, se  $m = 0$ , il corpo è vuoto e la clausola è detta clausola atomica o fatto, e tutte le clausole atomiche nella base di conoscenza sono sempre derivate in maniera diretta.

Nel caso del sistema di diagnostica del covid/influenza la base di conoscenza è stata organizzata dalle seguenti clausole:

**sintomi\_eguali**  $\Leftarrow$  mal di testa

**sintomi\_eguali**  $\Leftarrow$  mal di gola

**sintomi\_eguali**  $\Leftarrow$  dolori muscolari

**temperatura\_alta**  $\Leftarrow$  febbre

**rm\_polmoni**  $\Leftarrow$  esameok

**rm\_polmoni**  $\Leftarrow$  noesame  $\wedge$  tosse  $\wedge$  dolore\_al\_petto

**stanchezza**  $\Leftarrow$  tosse  $\wedge$  difficoltà\_di\_concentrazione  $\wedge$  febbre

**covid**  $\Leftarrow$  sintomi\_eguali  $\wedge$  rm\_polmoni  $\wedge$  temperatura\_alta

**influenza**  $\Leftarrow$  sintomi\_eguali  $\wedge$  stanchezza

La malattia può manifestarsi in due forme diverse, covid o influenza. Le due sono accomunate da dei sintomi uguali : **mal di testa, mal di gola e dolori muscolari**. I sintomi che caratterizzano il covid sono **risonanza magnetica ai polmoni positiva e temperatura alta, mentre l'influenza è caratterizzata dalla stanchezza**.

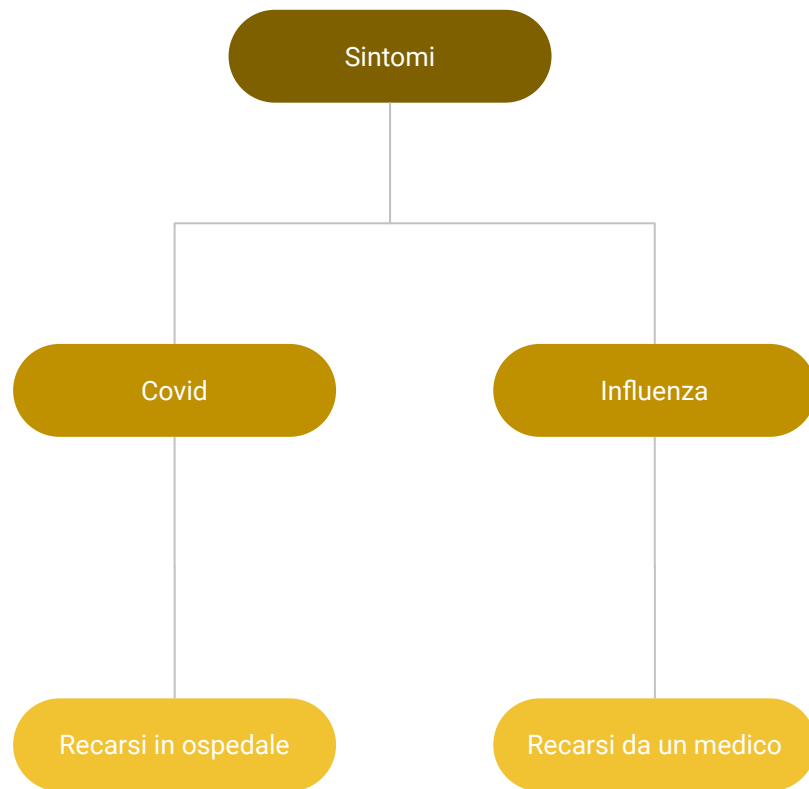
La possibilità di avere il covid può essere certo in caso in cui l'utente abbia sostenuto una risonanza magnetica con risultato positivo, oppure in caso di referto mancante il sistema tiene conto dei sintomi che possano rilevarne la presenza come la **tosse secca** e il **dolore al petto**. La presenza dell'influenza viene individuata secondo il riscontro dell'utente alla presenza di sintomi quali **difficoltà di concentrazione, tosse e febbre**.

# Sistema esperto

I **sistemi esperti** riproducono artificialmente le prestazioni di una persona esperta in un determinato dominio di conoscenza o campo di attività (motivo per il quale rientrano nella branca dell’Ai – Artificial Intelligence).

Un **sistema esperto** è in grado di mettere in atto autonomamente procedure di inferenza adeguate alla risoluzione di problemi particolarmente complessi.

Un **sistema esperto** è infatti un programma informatico che, dopo essere stato adeguatamente istruito, è capace di dedurre informazioni (output) da un insieme di dati e da informazioni di partenza (input).



# Implementazione sistema esperto

L'implementazione del sistema si basa su un sistema esperto realizzato tramite il linguaggio Python. Utilizziamo la libreria **Experta**, che permette di associare dei fatti accaduti a delle regole riguardo gli stessi. Le regole sono formate da due componenti chiamati **LHS** (Left-Hand-Side) e **RHS** (Right-Hand-Side). Il primo descrive le condizioni che si devono verificare affinché la regola venga applicata, mentre il secondo è l'insieme di azioni che vengono compiute quando viene applicata la regola.

Per ogni sintomo è stata associata una regola che, quando viene applicata, si occupa di domandare all'utente se riscontra il sintomo e, in base alla sua risposta (sì o no), il fatto relativo al sintomo viene impostato a true o a false, in seguito il sistema interpreterà la situazione dell'utente, e di conseguenza applicherà altre regole relative ad altri sintomi.

In base alla combinazione di fatti relativi ai sintomi impostati a true, possono essere richiamare altre regole, relative a sintomi correlati alla malattia. Alla fine, in base ai sintomi riscontrati dall'utente, viene applicata una regola tra quelle che si occupano di comunicare la diagnosi stabilita dal sistema quando richiamata.

Il sistema comincia a porre le domande all'utente relative ai sintomi di base. Se l'utente riscontra almeno uno tra i tre sintomi basilari (**mal di testa, mal di gola e dolori muscolari**), si prosegue con l'analisi cercando di stabilire di quale delle due malattie può essere affetto.

Il sistema andrà a verificare prima se si è affetti dal **corona virus**, controllando se l'utente è affetto da tutti i sintomi della malattia. Si chiede, quindi, all'utente se ha la **febbre** e se la **temperatura è alta**. Il sistema proseguirà andando a stabilire se l'utente ha effettuato un RM polmonare. Nel caso in cui si è sottoposto all'esame con esito positivo, la presenza di covid è certa, mentre nel caso di esito negativo, sarà nulla. Nel caso in cui l'utente non si è sottoposto a un esame, il sistema andrà a stabilire se soffre di **dolore al petto o se ha tosse secca**.

Se l'utente riscontra tutti i sintomi del covid, verrà fornita la diagnosi positiva. In mancanza di sintomi come tosse e dolore al petto, il sistema andrà a indagare riguardo la possibilità di essere affetti dall'influenza. Il sistema cercherà, così, di stabilire se soffre di stanchezza, sempre ponendogli delle domande riguardo l'aver riscontrato sintomi quali **tosse, difficoltà di concentrazione e febbre**. Se riscontra tutti e tre i sintomi allora il sistema stabilisce che l'utente soffre di **stanchezza** e di conseguenza comunicherà che ha **l'influenza**.

```
# SINTOMI eguali
@Rule(Fact(question=True))
def ask_mal_di_testa(self):
    self.declare(Fact(mal_di_testa=ask_question("Hai mal di testa ?")))

@Rule(Fact(question=True))
def ask_mal_di_gola(self):
    self.declare(Fact(mal_di_gola=ask_question("Hai mal di gola ?")))

@Rule(Fact(question=True))
def ask_dolori_muscolari(self):
    self.declare(Fact(dolori_muscolari=ask_question("Hai dolori muscolari ?")))
```

```
# febbre
@Rule(Fact(sintomi_eguali=True))
def ask_febbre(self):
    self.declare(Fact(febbre=ask_question("Hai la febbre?")))
```

Menù:

1) Sistema esperto

2) Esci

-----  
Seleziona (1)sistema esperto - (2)esci : 1

Benvenuto nel sistema diagnostico con modello a sistema esperto!

-----  
Hai dolori muscolari ? (si/no): si

Hai la febbre? (si/no): si

Hai la temperatura alta? (si/no): si

Hai svolto un RM ai polmoni per vedere se hai qualcosa? (si/no): no

Hai dolore al petto? (si/no): si

Hai tosse secca? (si/no): si

I sintomi indicano che potresti aver contratto il virus del covid.

È consigliabile recarsi in una struttura ospedaliera il prima possibile.

Premi un pulsante qualsiasi per continuare...■

Menù:

1) Sistema esperto

2) Esci

-----  
Seleziona (1)sistema esperto - (2)esci : 1

Benvenuto nel sistema diagnostico con modello a sistema esperto!

-----  
Hai dolori muscolari ? (si/no): no

Hai mal di testa ? (si/no): si

Hai la febbre? (si/no): si

Hai la temperatura alta? (si/no): no

Hai difficoltà a concentrarti? (si/no): si

Hai tosse secca? (si/no): si

I sintomi indicano che potresti avere un'influenza.

È consigliabile recarsi da un medico per ulteriori accertamenti o terapie.

Premi un pulsante qualsiasi per continuare...



# Sistema di predizione (Health system)

Abbiamo sviluppato una web app interattiva con **Streamlit** chiamata **Health System** che si occupa di effettuare previsioni con 4 dataset di dimensioni diverse riguardo 4 malattie dell'organismo, ovvero : parkinson, cancro ai polmoni, malattie del cuore, cancro alla prostata.

Di queste quattro malattie abbiamo effettuato la predizione tramite la **SVM** (Support Vector Machine).

Esempio (parkinson) :

MDVP:Fo (Hz)	MDVP:Fhi (Hz)	MDVP:Flo (Hz)	MDVP:Jitter (%)	MDVP:Jitter (Abs)
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer (dB)
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Shimmer:APQ3	Shimmer:APQ5	MDVP:APQ	Shimmer:DDA	NHR
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
HNR	RPDE	DFA	spread1	spread2
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
D2	PPE			
<input type="text"/>	<input type="text"/>			
<input type="button" value="Parkinson's Test Result"/>				
<div></div>				

MDVP:Fo (Hz)	MDVP:Fhi (Hz)	MDVP:Flo (Hz)	MDVP:Jitter (%)	MDVP:Jitter (Abs)
119.99200	157.30200	74.99700	0.00784	0.00007
MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer (dB)
0.00370	0.00554	0.01109	0.04374	0.42600
Shimmer:APQ3	Shimmer:APQ5	MDVP:APQ	Shimmer:DDA	NHR
0.02182	0.03130	0.02971	0.06545	0.02211
HNR	RPDE	DFA	spread1	spread2
21.03300	0.414783	0.815285	-4.813031	0.266482
D2	PPE			
2.301442	0.284654			
<input type="button" value="Parkinson's Test Result"/>				
<div>The person has Parkinson's disease</div>				

Split dei dati in training and test :

```
[13] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
[14] print(X.shape, X_train.shape, X_test.shape)
```

```
(195, 22) (156, 22) (39, 22)
```

```
[15] model = svm.SVC(kernel='linear')
```

```
[16] # training the SVM model with training data
      model.fit(X_train, Y_train)
```

```
model.fit(X_train,Y_train)
```

```
pred_x=pd.DataFrame(np.array([119.99200,157.30200,74.99700,0.00784,0.00007,0.00370,0.00554,0.01
```

```
prediction=model.predict(pred_x)
```

```
if (prediction[0] == 0):
    print("The Person does not have Parkinsons Disease")
```

```
else:
    print("The Person has Parkinsons")
```

Inoltre in ogni file relativo ad ogni malattia, abbiamo applicato la k-fold cross validation stratificata. Ci ha permesso di capire in base alla stampa data, le performance di quattro modelli diversi ( **KNN, SVM, Logistic Regression, Random Forest**).

**Es.** Se prendiamo in considerazione le malattie del cuore, abbiamo osservato che il modello con regressione logistica performa meglio rispetto agli altri, avendo uno score dell'accuratezza maggiore. Mentre se prendiamo in considerazione il cancro ai **polmoni, prostata o parkinson** risulterà che il modello più performante è quello del random forest.

**Random Forest** o Foresta casuale è un algoritmo di machine learning supervisionato. È uno degli algoritmi più usati grazie alla sua accuratezza, semplicità e flessibilità. Il fatto che possa essere usato per compiti di classificazione e regressione, unito alla sua natura non lineare, lo rende altamente adattabile a una gamma di dati e situazioni. Si chiama "foresta" perché fa crescere una foresta di alberi di decisione. I dati di questi alberi vengono poi fusi insieme per assicurare le previsioni più accurate. Mentre un albero decisionale da solo dispone di un risultato e una gamma ristretta di gruppi, la foresta assicura un risultato più accurato con un numero maggiore di gruppi e decisioni. Ha l'ulteriore vantaggio di aggiungere casualità al modello trovando la caratteristica migliore tra un sottoinsieme casuale di caratteristiche. Nel complesso, questi benefici creano un modello con un'ampia diversità che molti data scientist privilegiano.

E' possibile visualizzare dalla stampa quale fra i tre modelli performa meglio.

(cancro alla prostata)

```
SVM: 0.816667 (0.116667)  
RF: 0.850000 (0.116667)  
LR: 0.833333 (0.129099)  
KNN: 0.766667 (0.110554)
```

(malattie del cuore)

```
SVM: 0.611500 (0.061094)  
LR: 0.826667 (0.067696)  
KNN: 0.636167 (0.114746)  
RF: 0.797667 (0.080654)
```

(parkinson)

```
SVM: 0.813333 (0.054924)  
LR: 0.846250 (0.071065)  
KNN: 0.845833 (0.079342)  
RF: 0.929583 (0.052785)
```

(cancro ai polmoni)

```
SVM: 0.881349 (0.015918)  
RF: 0.924603 (0.033521)  
LR: 0.917196 (0.028179)  
KNN: 0.870238 (0.051856)
```

Nella web app (**Health System**) inoltre è possibile visualizzare la predizione del cancro al seno. L'utente può selezionare il tipo di classificatore (**KNN, SVM, Random Forest**).

Select classifier

KNN

KNN

SVM

Random Forest

Health system

All dataset used

Parkinsons prediction

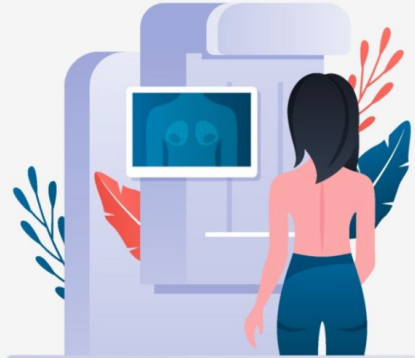
Lung Cancer prediction

Heart Disease prediction

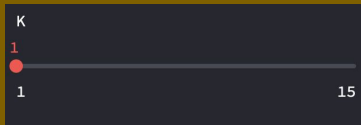
Prostate Cancer prediction

Breast Cancer classifier

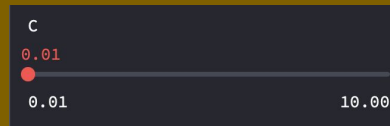
# Breast Cancer Classifier



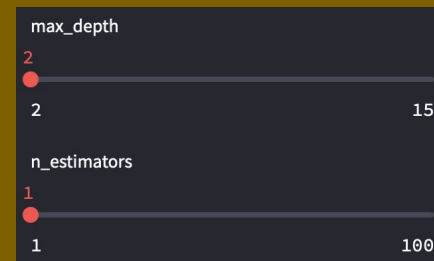
Nello specifico il **KNN o k-nearest neighbors**, permette di selezionare il k ( che varia tra 1 e 15 ) ed in base al valore scelto cambia il risultato dell'accuratezza. ( k indica il numero di elementi vicini nello spazio da considerare).



Nel caso in cui selezioniamo l'**SVM** o support vector machine, permette di selezionare il **gamma** parametro C ( che varia tra 0.01 e 10.00 ) ed in base al valore scelto cambia il risultato dell'accuratezza. (C è essenzialmente un parametro di regolarizzazione, che controlla il compromesso tra il raggiungimento di un errore basso sui dati di addestramento e la riduzione al minimo della norma dei pesi. Aumentando C, aumentiamo la complessità della classe di ipotesi).



Infine nel caso in cui selezioniamo il **Random Forest**, possiamo anche scegliere il valore della profondità massima(max\_depth - che varia tra 2 e 15 ) e del numero di estimatori (n\_estimators - che varia tra 1 e 100) ed in base al valore scelto cambia il risultato dell'accuratezza.



Nella web app (Health System) infine è possibile visualizzare ogni dataset usato, caricandolo da locale.


### Upload Dataset

Upload CSV file

Drag and drop file here


Limit 200MB per file • CSV


Browse files


 **breastCancer.csv** ×


125.2KB


### Health system


 -----  
**All dataset used**

 -----  
Parkinsons prediction

 -----  
Lung Cancer prediction

 -----  
Heart Disease prediction

 -----  
Prostate Cancer prediction

 -----  
Breast Cancer classifier

## DATASET

### View of the loaded dataset

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.9900	10.3800	122.8000	1,001.0000	0.1184
1	842517	M	20.5700	17.7700	132.9000	1,326.0000	0.0847
2	84300903	M	19.6900	21.2500	130.0000	1,203.0000	0.1096
3	84348301	M	11.4200	20.3800	77.5800	386.1000	0.1425
4	84358402	M	20.2900	14.3400	135.1000	1,297.0000	0.1003
5	843786	M	12.4500	15.7000	82.5700	477.1000	0.1278
6	844359	M	18.2500	19.9800	119.6000	1,040.0000	0.0946

# Implementazione

Abbiamo sviluppato l'health system inizialmente in google collab, creando ogni singolo file sulle 4 malattie diverse. La libreria utilizzata per la predizione è sklearn. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano e k-mean ed è progettato per operare con le librerie NumPy e SciPy.

E' una libreria open-source di apprendimento automatico per il linguaggio di programmazione python. Ci ha permesso di analizzare il dataset e effettuare le varie predizioni con classificatori diversi.

Dopo aver suddiviso ogni dataset e analizzato al meglio, abbiamo importato i file in un unico file python usando la libreria pickle. Il modulo pickle implementa un basilare ma potente algoritmo, per serializzare e deserializzare una struttura di oggetti Python.

Nel file python(in spyder) abbiamo creato l'interfaccia grafica su streamlit e implementato il tumore al seno sviluppando la possibilità di far scegliere il classificatore all'utente.