

Documentazione progetto Ingegneria della Conoscenza

Indice

1. [Introduzione](#)
2. [Requisiti Funzionali](#)
3. [Manuale utente](#)
4. [Scelte progettuali](#)
5. [Architettura del sistema](#)
6. [Implementazioni future](#)
7. [Processo e sviluppo dell'organizzazione del lavoro](#)
8. [Conclusioni](#)

Introduzione

Il sistema è in grado di prevedere se si è positivi al virus SARS-CoV-2 a seconda dei sintomi analizzati dal dataset preso in considerazione e inoltre è in grado di consigliare all'utente se rivolgersi ad uno specialista, spiegheremo meglio più avanti.

Requisiti Funzionali

I requisiti funzionali sono relativamente pochi.

Avendo progettato il programma in Python, si richiede un ambiente in grado di eseguire codice come Eclipse, IntelliJ...

Relativamente alle librerie esterne importate, è necessario installare sulla macchina:

- **pandas**, usato per la manipolazione e l'analisi dei dati;
- **scikit-learn**, per applicare i concetti del Machine Learning;
- **numpy**, per lavorare con array multidimensionali e applicare specifiche operazioni logiche/matematiche;
- **pytholog**, per utilizzare la programmazione logica in Python.

Ognuna delle quali può essere installata dal CLI di windows con il comando:

`"python -m pip install nome_libreria"`

Manuale utente

Dopo averlo scaricato si dovrà eseguire il file "prediction" su un IDE.

```

Benvenuto nel sistema per predire se sei affetto dal virus SARS-CoV-2
Per capire cio' vi faremo una serie di domande su patologie pregresse e sintomi che potreste avere
Vi preghiamo di rispondere con la massima sincerita'
Grazie

Le risposte devono essere date nei possibili seguenti modi:
->si-s=yes-y
->no-n
Hai problemi di respirazione?
|

```

Questo rappresenta la schermata iniziale una volta avviato correttamente il programma, presenta una breve descrizione e poi iniziano le domande necessarie ad effettuare la predizione.

Qui di fianco troviamo appunto le domande richieste, che verranno richieste sequenzialmente. In particolare in questa foto si è presentato uno dei casi in cui il sistema consiglia la visita specialistica a causa dei sintomi comunicati.

```

Hai problemi di respirazione?
si
Hai febbre?
n
Hai tosse secca?
si
Hai mal di gola?
no
Hai asma?
si
Hai malattia del cuore?
no
Hai diabete?
no
Hai ipertensione?
si
Hai viaggiato all'estero?
no
Hai avuto contatti con pazienti COVID?
no
Hai partecipato ad un grande raduno?
n
Hai visitato posti esposti al pubblico?
s
La tua famiglia lavora in luoghi esposti al pubblico?
n
Si consiglia la visita da uno specialista!

```

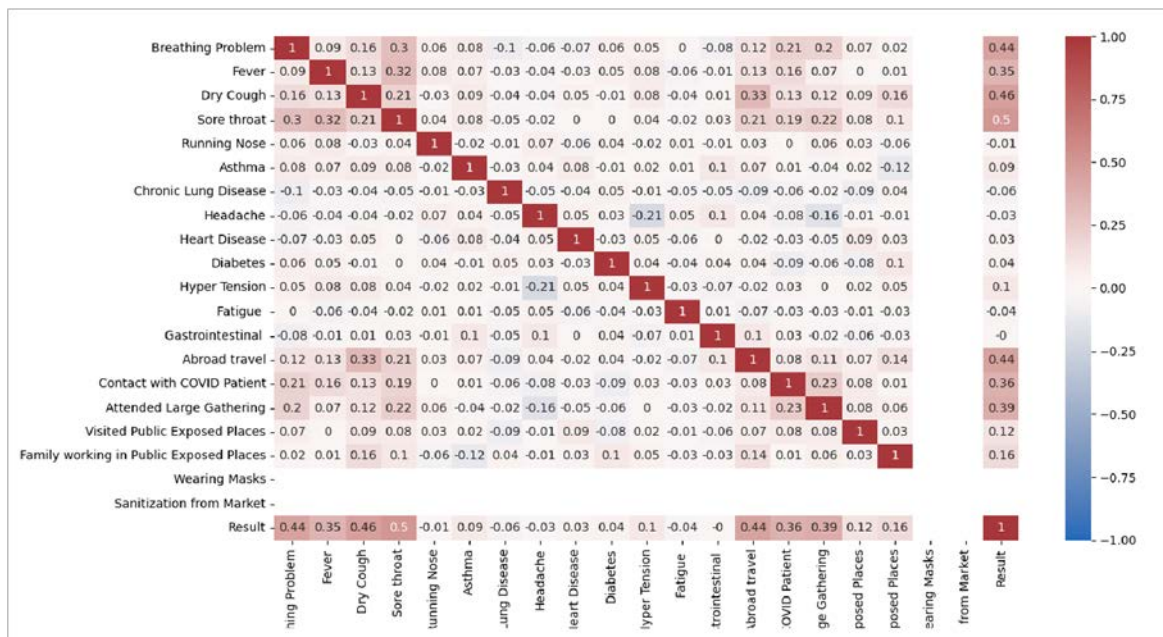
Secondo le prestazioni del sistema e le risposte date, l'utente non è affetto da Covid-19

Scelte progettuali

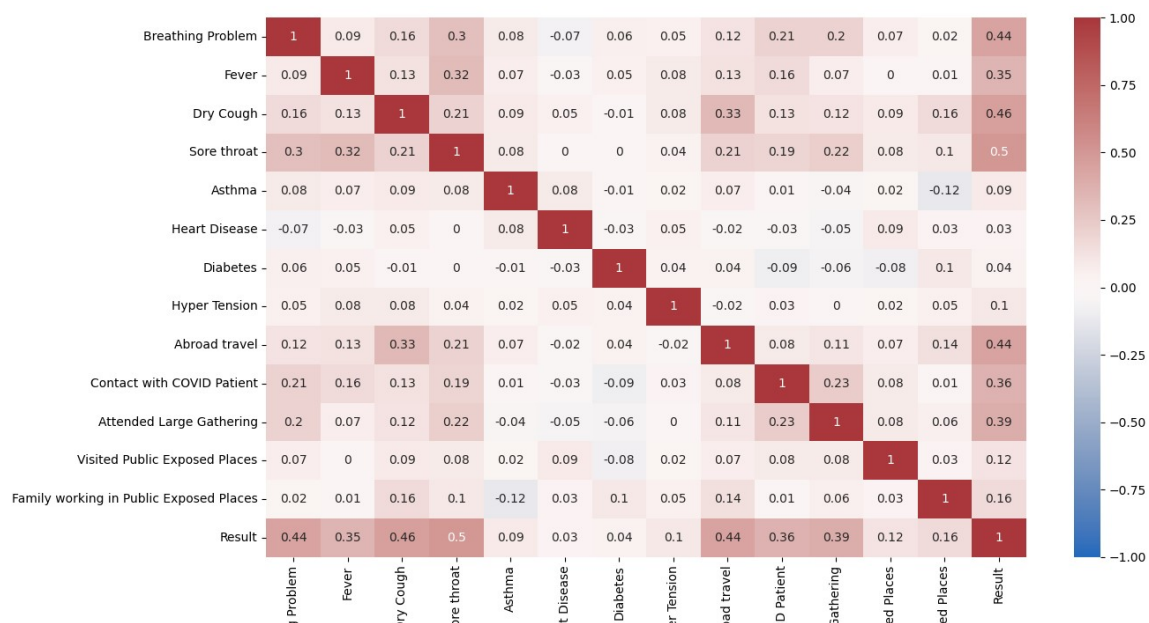
Si è utilizzato il dataset dal seguente link: [Symptoms and COVID Presence \(May 2020 data\)](https://www.kaggle.com/robertogarcia/symptoms-and-covid-presence) | [Kaggle](https://www.kaggle.com/robertogarcia/symptoms-and-covid-presence) per addestrare il sistema.

I modelli e le tecniche utilizzate:

- prima di passare direttamente all'apprendimento supervisionato, è stato osservato il dataset scelto, e mediante una matrice di correlazione è stato notato che alcune features risultavano poco coerenti/discriminanti nei confronti della feature obiettivo e pertanto si è effettuato *features selection*, passando da questa prima situazione



a quest'altra



```
data = pd.read_csv(r"..\dataset\usable.csv")
```

```
data=data.drop('Wearing Masks',axis=1)
data=data.drop('Sanitization from Market',axis=1)
data=data.drop('Running Nose',axis=1)
data=data.drop('Chronic Lung Disease',axis=1)
data=data.drop('Headache',axis=1)
data=data.drop('Fatigue ',axis=1)
data=data.drop('Gastrointestinal ',axis=1)
data.to_csv(r"..\dataset\usable.csv", index=False)
```

calcolando accuratezza del

sistema prima e dopo l'eliminazione delle features si è notato una leggera diminuzione della precisione (circa 0.02), ma che si è pensato fosse accettabile, all'ulteriore fine di non stancare l'eventuale utente durante l'interazione e quindi ricevere risposte reali e non date a caso.

Per plottare la matrice di correlazione, si è usato un metodo di Pandas, in particolare `pandas.DataFrame.corr()`, e come metrica di correlazione Pearson.

- Si è utilizzato un albero decisionale come classificatore, per la predizione della feature obiettivo.

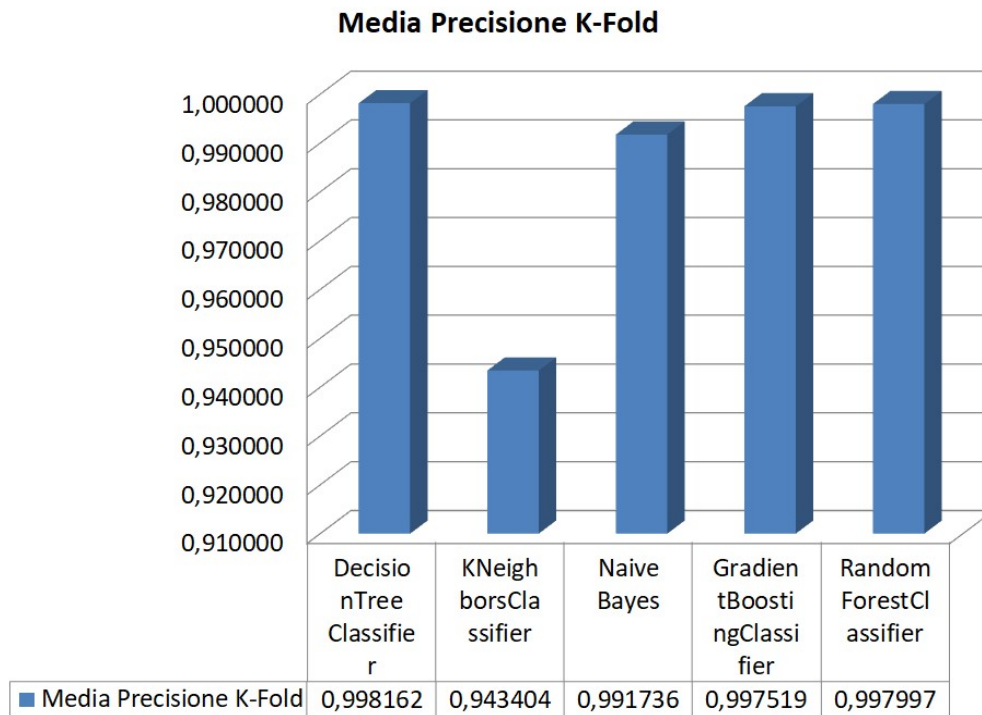
Questa scelta non è stata casuale ma mi sono basato sui seguenti dati che mostrano l'accuratezza riscontrata in 5 esecuzioni del sistema, ognuna delle quali utilizzando una K-Fold cross validation, mostra la media riscontrata tra le precisioni calcolate nelle varie convalide, *deviation* indica la deviazione standard:

Model	K-Fold	Deviation
<i>DecisionTree Classifier</i>	0,998100	0,001
<i>DecisionTree Classifier</i>	0,998085	0,001
<i>DecisionTree Classifier</i>	0,998232	0,000
<i>DecisionTree Classifier</i>	0,998224	0,001
<i>DecisionTree Classifier</i>	0,998169	0,001
Media	0,998162	0,001
<i>KNeighborsClassifier</i>	0,946329	0,001
<i>KNeighborsClassifier</i>	0,942096	0,001
<i>KNeighborsClassifier</i>	0,940061	0,002
<i>KNeighborsClassifier</i>	0,945248	0,002
<i>KNeighborsClassifier</i>	0,943285	0,002
Media	0,943404	0,001
<i>Naive Bayes</i>	0,991677	0,005
<i>Naive Bayes</i>	0,991738	0,002
<i>Naive Bayes</i>	0,991681	0,002
<i>Naive Bayes</i>	0,991951	0,003
<i>Naive Bayes</i>	0,991633	0,003
Media	0,991736	0,003

<i>GradientBoostingClassifier</i>	0,997600	0,001
<i>GradientBoostingClassifier</i>	0,997553	0,001
<i>GradientBoostingClassifier</i>	0,997414	0,001
<i>GradientBoostingClassifier</i>	0,997347	0,002
<i>GradientBoostingClassifier</i>	0,997683	0,001
Media	0,997519	0,001
<i>RandomForestClassifier</i>	0,998026	0,001000
<i>RandomForestClassifier</i>	0,998103	0,001000
<i>RandomForestClassifier</i>	0,998300	0,001000
<i>RandomForestClassifier</i>	0,997290	0,001000
<i>RandomForestClassifier</i>	0,998264	0,000000
Media	0,997997	0,001

Questo classificatore è stato scelto a causa delle alte prestazioni rispetto agli altri e non si è preso affatto in considerazione i regressori a causa del contenuto del dataset, che contiene soli valori discreti, in particolare binari.

Il file di cui è stato mostrato l'estratto è disponibile nel repository fornito, al seguente percorso [test_models](#), del quale di seguito un breve confronto



- Come strumento per misurare l'accuratezza del sistema, mi sono avvalso della K-Fold cross-validation. In particolare usavo 10 fold e come metrica per l'accuratezza la "Compute Area Under the Receiver Operating Characteristic Curve" (*roc_auc*), si è scelto questa a causa della sua compatibilità con features binarie. Si è implementato anche una funzione che permette di visionare la precisione del sistema, nell'esecuzione corrente. Anche qui come metrica per la precisione si è utilizzata la *roc_auc*, che utilizza di default una macro-media.
- così come per l'accuratezza, si è pensato potesse tornare utile tenere conto dell'errore che il sistema presenta, quindi sempre mediante un'apposita funzione, è possibile osservare il "Mean Absolute Error", cioè l'errore assoluto riscontrato tra le features predette e quelle reali sia in fase di training che di test.

```

Accuracy
L'accuratezza sulle predizioni fatte in fase di apprendimento e' di 0.998462794307433
L'accuratezza sulle predizioni fatte in fase di test e' di 0.9992479197781097

Mean Absolute Error
MAE train 0.02362294538408
MAE test 0.017940603129195588
  
```

- Mi sono cimentato con prolog, effettuando una semplice KB, composta da asserzioni che specificano la manifestazione, o meno, di specifici sintomi comunicati dall'utente, così da consigliare di andare da uno specialista per una visita più approfondita, cioè a prescindere dalla predizione effettuata dal sistema. In particolare viene consigliata una visita se l'utente comunica di avere *asma*, *problemi respiratori* e *tosse secca* o *bruciore di gola*, il che potrebbe potenzialmente manifestare una complicazione del quadro clinico del paziente, sempre relativamente al covid. Per fare ciò si è inserito in una KB, 2 proposizioni: *symp* e *several*. La prima permette di asserire la presenza del sintomo, a seconda delle risposte dell'utente, la seconda

verrà utilizzata in fase di interrogazione, in particolare restituirà valore vero in due possibili casi:

- a. l'utente ha dichiarato di avere asma, problemi respiratori e tosse secca;
- b. l'utente ha dichiarato di avere asma, problemi respiratori e bruciore di gola;

```
kb([f"symp(asma,{ris[0][4].Lower()})",
    f"symp(breathproblem,{ris[0][0].Lower()})",
    f"symp(sorethroat,{ris[0][3].Lower()})",
    f"symp(drycough,{ris[0][2].Lower()})",
    "several(S) :- symp(asma, S), symp(breathproblem, S), symp(drycough, S)",
    "several(S) :- symp(asma, S), symp(breathproblem, S), symp(sorethroat, S)",
    ])
```

Architettura del sistema

Il sistema presenta la seguente struttura:

- *esame*
 - *dataset*, contenente i dataset utilizzati. In particolare, *covid_symptoms* è quello originale, cioè così come reperito da Internet, che si è pensato fosse bene conservare, e il secondo *usable* che rappresenta il dataset effettivamente utilizzato in fase di apprendimento;
 - *src*, contenente tutto il codice sorgente.
 - *model_performance.py*, contiene le funzioni utili a osservare le prestazioni del sistema (accuratezza, K-Fold e l'errore assoluto). Per quanto riguarda queste funzioni, si è deciso di implementarle, non renderle utilizzabili all'utente ma conservarle per applicazioni/modifiche future, in cui possano tornare utili.
 - *prediction.py*, costituisce il cuore del progetto, si occupa di richiamare le librerie necessarie, aprire il dataset, creare un modello, addestrare il sistema e in seguito al richiamo dei metodi necessari all'acquisizione dei sintomi dell'utente, effettuare la predizione vera e propria.
 - *prolog.py*, contiene la base di conoscenza che include i sintomi più gravi che possono suggerire una visita da uno specialista
 - *questions.py*, contiene la parte di interazione con l'utente in cui vengono poste una serie di domande;
 - *user.py*, una classe con metodi e attributi necessari ad asserire i sintomi comunicati dall'utente;

Implementazioni future

Avrei voluto implementare altre funzionalità in modo da migliorarlo ancora, ma a causa delle limitate conoscenze pratiche, mi sono dovuto fermare.

In particolare avrei voluto implementare la funzionalità per cui una volta ottenuta la predizione, se pari ad 1, quindi potenzialmente positiva al covid, chiedendo la nazione di

residenza, avrei voluto predire la variante del virus, utilizzando un dataset trovato in rete fornito da un ente [autorevole](#) e pertanto affidabile.

Oltre questo sarei stato interessato a prevedere una data indicativa, relativa all'assenza di casi Covid-19, per il mondo intero sarebbe difficile, avrei potuto considerare una singola nazione, osservando dataset sui casi giornalieri e dataset sulle vaccinazioni.

Conclusioni

Il presente progetto si è concentrato sulla predizione della presenza del virus covid negli utenti. Attraverso un'analisi approfondita di dati clinici e sintomi correlati, si è sviluppato un modello predittivo che dimostra promettenti risultati nella sua capacità di discriminare tra casi positivi e negativi con una significativa precisione.

Sono stati esaminati diversi modelli di apprendimento automatico, tra cui il Decision Tree Classifier, il K-Nearest Neighbors, il Naive Bayes, il Gradient Boosting e il Random Forest. Attraverso un'attenta valutazione e confronto dei risultati ottenuti da ciascun modello, è emerso che il Random Forest ha dimostrato la migliore performance tra tutti.