

**Autori: Simone Della Porta, Antonio De Lucia,
Antonio Maddaloni, Rocco Iuliano, Francesco Peluso**

Repository GitHub: [link](#)

- 1. Business Understanding**
 - 1.1 Introduzione al problema**
 - 1.2 Obiettivi di business**
 - 1.3 Descrizione dell'ambiente**
 - 1.3.1 Specifica PEAS**
 - 1.4 Business Success Criteria**
 - 1.5 Tool da utilizzare**
- 2. Data Understanding**
 - 2.1 Scelta del dataset**
 - 2.2 Analisi del dataset**
 - 2.3 Data quality**
 - 2.4 Data Exploration**
- 3. Data Preparation**
 - 3.1 Data Cleaning**
 - 3.2 Define data and target variables**
 - 3.3 Data Balancing**
 - 3.4 Feature Scaling**
 - 3.5 Feature Engineering**
- 4. Data Modeling**
- 5. Evaluation**

1. Business Understanding

1.1 Introduzione al problema

Con l'avvenire della pandemia del covid-19 iniziata nel 2019 molte organizzazioni sanitarie stanno cercando di tener traccia dei contagi e del tasso di mortalità dovuti ad essa; tutto ciò per limitare l'aumento della curva dei contagi.

1.2 Obiettivi di business

Dato il problema in esame, l'obiettivo del nostro progetto è realizzare un modello di machine learning che sia capace di predire se una persona risulta essere positiva o negativa al covid-19 sulla base di alcuni sintomi.

1.3 Descrizione dell'ambiente

1.3.1 Specifica PEAS

PEAS	
Performance	La misura di performance dell'agente è la sua capacità di avvicinarsi quanto più possibile ad una situazione ideale nella quale vengono predetti correttamente i veri positivi e negativi al covid-19.
Environment	L'ambiente in cui opera il nostro agente riguarda l'ambito sanitario e più in particolare la virologia. L'ambiente è: Statico in quanto l'ambiente non cambia nel tempo e non cambia alle azioni effettuate dell'agente. Episodico in quanto un'azione intrapresa dall'agente in un dato istante non è influenzata dall'azione effettuata precedentemente. Completamente osservabile in quanto l'agente in ogni istante può avere una visione completa dell'ambiente in cui è calato. Discreto in quanto l'agente può effettuare soltanto determinate azioni e ricevere determinati impulsi. Non noto in quanto l'agente non conosce il risultato delle proprie azioni. Singolo in quanto l'ambiente prevede al proprio interno l'introduzione di un singolo agente. Gli elementi dell'ambiente sono le persone che hanno determinati sintomi e che possono essere possibili positivi al covid-19.
Actuators	L'agente agisce sull'ambiente tramite lo stream di output del nostro computer nel quale andrà a predire la positività di una persona.
Sensors	L'agente riceverà gli impulsi tramite lo stream di input del nostro computer.

1.4 Business success criteria

Il criterio con il quale andremo a validare il nostro sistema è il seguente:

Il modello deve avere un buon livello di accuracy, ovvero almeno il 50%. Questo perché essendo un problema di natura medica e trattiamo un virus nuovo, vuol dire che i dati a disposizione per far apprendere il modello non sono sicuramente accurati. Inoltre è da notare che il modello predice la positività o meno di una persona in base ai sintomi e non è detto che se una persona ha determinati sintomi allora ha il covid, ma potrebbe avere altre patologie che comportano sintomi simili. Un ulteriore motivo per il quale abbiamo scelto questa soglia di accuracy è che il modello potrà sbagliare molto facilmente con persone asintomatiche in quanto non hanno sintomi e risultano come persone “negative” agli “occhi” del modello.

1.5 Tool da utilizzare

I tool da utilizzare per sviluppare il progetto sono:

- Visual Studio Code
- Python
- Pandas
- GitHub
- Scikitlearn
- Mathplot
- Kaggle
- Overleaf

2. Data Understanding

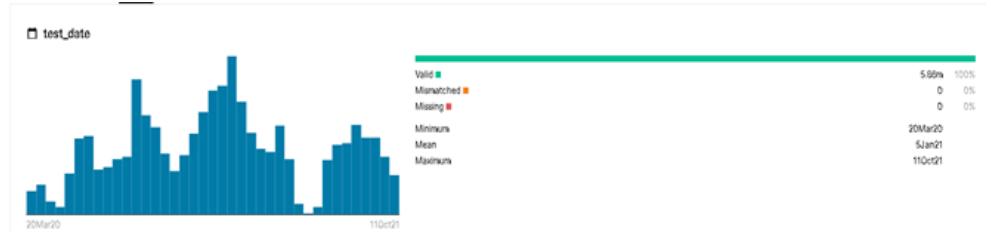
2.1 Scelta del dataset

Per la creazione del nostro machine learner, abbiamo analizzato vari dataset in rete e abbiamo deciso che quello più opportuno alle nostre esigenze e al raggiungimento dei nostri obiettivi è il seguente:

[Click This](#)

2.2 Analisi del dataset

- Nel dataset in questione i dati sono stati collezionati nel lasso di tempo Marzo 2020-Novembre 2021, i dati sono stati reperiti dal sito del governo di Israele ([Click This](#)).
- Nel dataset sono presenti 10 colonne che sono:
 - **test_data**: rappresenta la data di esecuzione del test. La data è espressa nel formato americano, quindi yyyy-mm-dd. Riportiamo l'analisi di questa caratteristica:



- **cough**: indica se la persona presenta il sintomo della tosse. Viene espresso sotto forma di numero binario (1 ha la tosse e 0 non ha la tosse). Riportiamo l'analisi di questa caratteristica:



- **fever**: indica se la persona presenta il sintomo della febbre. Viene espresso sotto forma di numero binario (1 se ha la febbre e 0 se non ha la febbre). Riportiamo l'analisi di questa caratteristica:



- **sore_throat:** indica se la persona presenta il sintomo del mal di gola. Viene espresso sotto forma di numero binario (1 se ha mal di gola e 0 se non ha mal di gola). Riportiamo l'analisi di questa caratteristica:



- **shortness_of_breath:** indica se la persona presenta problemi respiratori ovvero fiato corto. Viene espresso sotto forma di numero binario (1 se ha il fiato corto e 0 se non ha il fiato corto). Riportiamo l'analisi di questa caratteristica:



- **head_ache**: indica se la persona presenta mal di testa. Viene espresso sotto forma di numero binario (1 se ha mal di testa e 0 se non ha mal di testa). Riportiamo l'analisi di questa caratteristica:



- **corona_result**: indica se la persona è positiva o negativa al covid. Viene espresso sotto forma di categoria (positivo o negativo). In questo dataset questa caratteristica rappresenta la **variabile dipendente**. Riportiamo l'analisi di questa caratteristica:



- **age_60_and_above**: indica se la persona ha più di 60 anni o meno di 60 anni. Viene espresso sotto forma booleana (yes o no). Riportiamo l'analisi di questa caratteristica:



- **gender**: indica se la persona è maschio o femmina. Viene espresso sotto forma di categoria (male o female). Riportiamo l'analisi di questa caratteristica:



- **test_indication**: indica se la persona ha avuto possibili contatti con persone positive. Viene espresso sotto forma di categoria:
 - * **contact with confirmed** indica che la persona ha avuto un contatto con una persona positiva;
 - * **other** indica che la persona non è sicura di aver avuto contatti con positivi;
 - * **above** indica che la persona non è sicura di aver avuto contatti con positivi ma torna da un viaggio all'estero.

Riportiamo l'analisi di questa caratteristica:



Nel dataset sono presenti 5.86 milioni di righe e il dataset ha la dimensione di 273mb.

2.3 Data quality:

Nel dataset non ci sono dati mancanti e la maggior parte dei dati sono rappresentati sulla stessa scala. C'è bisogno di prestare maggiore attenzione ad alcune caratteristiche come test_date e test_indication nella fase di data preparation. Questo perché test_date e test_indication hanno un formato e una scala diversa.

2.4 Esplorazione:

non sono state individuate correlazioni tra dati.

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

cough      fever      sore_throat      shortness_of_breath      head_ache
count  5.861480e+06  5.861480e+06  5.861480e+06  5.861480e+06
mean   4.676498e-02  3.989388e-02  1.891724e-02  4.552775e-03  3.923685e-02
std    2.111351e-01  1.938184e-01  1.362328e-01  6.732049e-02  1.941580e-01
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%   0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%   0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%   0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max   1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

analisi test date
count      5861480
unique      377
top       2021-01-04
freq      37493
Name: test_date, dtype: object

analisi gender
count      5861480
unique      2
top       female
freq      3086361
Name: gender, dtype: object

analisi test_indication
count      5861480
unique      3
top       Other
freq      5359282
```

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

top      female
freq    3886361
Name: gender, dtype: object

analisi test_indication
count    5861488
unique     3
top      Other
freq    5359282
Name: test_indication, dtype: object
test_date      0
cough         0
fever         0
sore_throat   0
shortness_of_breath  0
head_ache     0
corona_result  0
age_60_and_above  0
gender        0
test_indication  0
dtype: int64
PS C:\Users\39338\git\PROGETTAML> |
```

3. Data Preparation

3.1 Data Cleaning:

dalla precedente analisi dei dati abbiamo notato che non ci sono dati mancanti o rumorosi e quindi non utilizzeremo nessuna tecnica o algoritmo di data imputation.

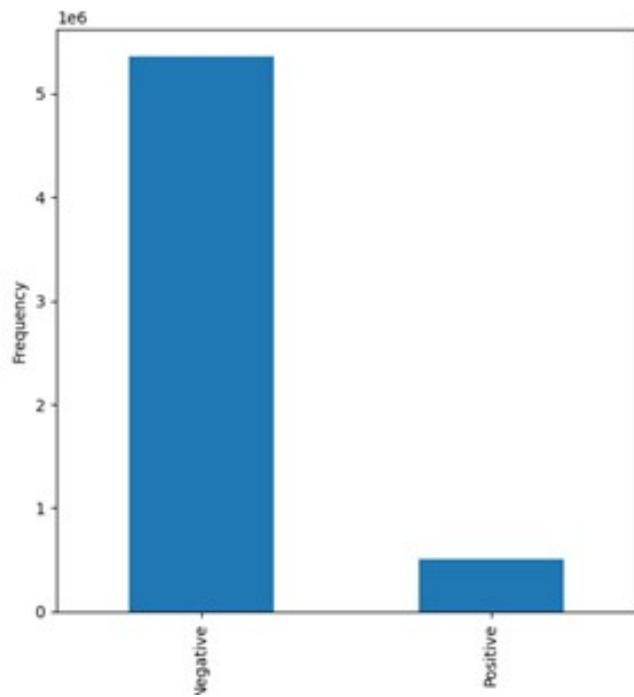
3.2 Define data and Target Variables:

la variabile target del nostro dataset è la variabile “corona_result” ed è la settima colonna del dataset

3.3 Data Balancing:

plotando i dati su un grafico a barre abbiamo notato che il nostro dataset è sbilanciato in quanto presenta molti casi negativi rispetto a quelli positivi. Infatti questi ultimi sono solo 505673 mentre le restanti 5355807 righe sono rappresentate dai negativi.

```
Negative      5355807
Positive      505673
Name: corona_result, dtype: int64
```



Per bilanciare il dataset utilizzeremo la tecnica di Undersampling perché avendo a disposizione molte righe non abbiamo il problema della sottodimensionalità dei dati. Non utilizziamo Oversampling perché la differenza tra il numero di istanze tra la classe di maggioranza e quella di minoranza è notevole e quindi facendo Oversampling aumenterebbe la probabilità di generare fake data.

Prima di effettuare l'Undersampling abbiamo diviso il dataset in test set e training set con le seguenti proporzioni: 30% e 70% rispettivamente. Questo perché non bisogna mai bilanciare il test set altrimenti andremo a falsificare il problema in analisi. Il numero di istanze ottenute per i set sono:

```
Number transactions X_train dataset: (4103036, 9)
Number transactions y_train dataset: (4103036,)
Number transactions X_test dataset: (1758444, 9)
Number transactions y_test dataset: (1758444,)
```

Con:

- X_train indichiamo le variabili indipendenti del training set
- y_train indichiamo la variabile dipendente del training set
- X_test indichiamo le variabili indipendenti del test set
- y-test indichiamo la variabile dipendente del test set

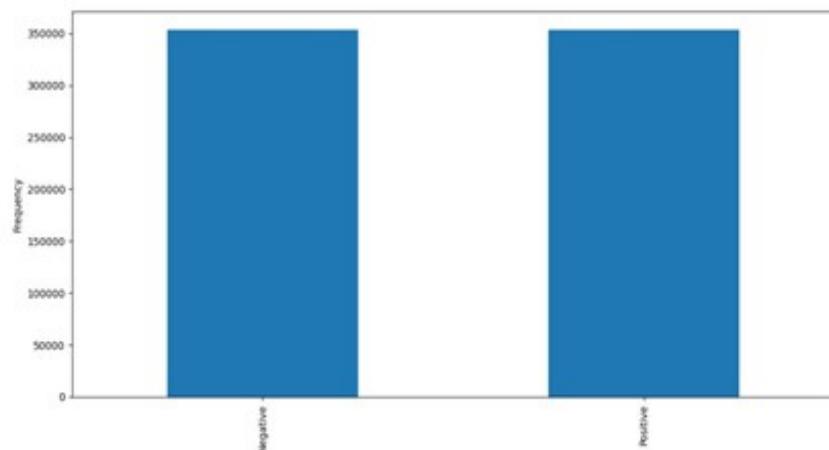
L'algoritmo utilizzato per effettuare l'Undersampling è il RandomUnderSampler. La situazione prima di applicare l'algoritmo sul training set era la seguente:

```
Prima UnderSampling, counts of label 'Positive': 353869
Prima UnderSampling, counts of label 'Negative': 3749167
```

La situazione dopo aver applicato l'algoritmo sul training set è la seguente:

```
Dopo UnderSampling, counts of label 'Positive': 353869
Dopo UnderSampling, counts of label 'Negative': 353869
```

Plotando il dataset di train bilanciato otteniamo il seguente grafico:



Durante l'addestramento e il test del modello, verificheremo i risultati ottenuti sia applicando il data balancing e sia senza applicarlo.

3.4 Feature Scaling:

All'interno del nostro dataset sono presenti feature con scala diversa, questo potrebbe portare un problema al machine learner in quanto potrebbe mal interpretare i dati e considerare una feature più importante di un'altra. Inoltre, il nostro dataset contiene il campo test_date che potrebbe risultare ambiguo al machine learner poiché non spiega alla perfezione quello che è il nostro fenomeno che stiamo analizzando. Quindi per dare maggior interpretabilità abbiamo scalato la data in base alla curva dei contagi, infatti, abbiamo notato che la curva tende a crescere nei mesi da gennaio a maggio e da ottobre a dicembre, mentre tende a decrescere da giugno a settembre. Le feature che abbiamo normalizzato sono le seguenti:

- **Gender** (abbiamo sostituito male=1 e female=0)
- **Test_date** (le date comprese tra gennaio e maggio e da ottobre a dicembre le abbiamo sostituite con 1 mentre le date da giugno ad agosto con 0)
- **Test_indication** (abbiamo accorpato Other e Abroad con valore 0 in quanto entrambi i valori esprimono l'incertezza della persona se ha avuto contatti con un positivo, mentre Contact with confirmed con 1)
- **Age_60_and_above** (abbiamo sostituito Yes=1 e No=0)

Abbiamo riportato tutte le feature su scala binaria proprio perché la maggior parte delle caratteristiche all'interno del nostro dataset sono sulla medesima scala.

```

#FEATURE SCALING
#andiamo a normalizzare la distribuzione delle seguenti feature: test_date, test_indication, gender, age_60_and_above
print("Inizio Feature Scaling")
#normalizziamo gender in questo modo: male->1 , female->0
print("Normalizzo gender...")
x_train_res["gender"] = x_train_res["gender"].replace({"male":1})
x_train_res["gender"] = x_train_res["gender"].replace({"female":0})

#lo applichiamo anche sul test set
x_test["gender"] = x_test["gender"].replace({"male":1})
x_test["gender"] = x_test["gender"].replace({"female":0})

#normalizziamo age_60_and_above in questo modo: yes->1 , no->0
print("Normalizzo age_60_and_above...")
x_train_res["age_60_and_above"] = x_train_res["age_60_and_above"].replace({"Yes":1})
x_train_res["age_60_and_above"] = x_train_res["age_60_and_above"].replace({"No":0})

x_test["age_60_and_above"] = x_test["age_60_and_above"].replace({"Yes":1})
x_test["age_60_and_above"] = x_test["age_60_and_above"].replace({"No":0})

#normalizziamo test_indication andando ad unire i seguenti 2 valori in un unico valore: Other, Abroad
#la normalizzazione verrà fatta nel seguente modo: other,abroad->0 , Contact with confirmed->1
print("Normalizzo test_indication...")
x_train_res["test_indication"] = x_train_res["test_indication"].replace({"Other":0})
x_train_res["test_indication"] = x_train_res["test_indication"].replace({"Abroad":0})
x_train_res["test_indication"] = x_train_res["test_indication"].replace({"Contact with confirmed":1})

x_test["test_indication"] = x_test["test_indication"].replace({"Other":0})
x_test["test_indication"] = x_test["test_indication"].replace({"Abroad":0})
x_test["test_indication"] = x_test["test_indication"].replace({"Contact with confirmed":1})

```

```

#normalizziamo test_date in base alla curva dei contagi
#consideriamo i mesi in cui la curva dei contagi e' alta con valore 1 ovvero: gennaio-maggio e da ottobre-dicembre
#consideriamo i mesi in cui la curva dei contagi e' bassa con valore 0 ovvero: giugno-settembre
print("Normalizzo test_date...")
x_train_res= x_train_res.reset_index()
y_train_res= y_train_res.reset_index()
i=0
#normalizziamo le date del training set
for row in x_train_res.iterrows():
    if row.test_date>="2020-01-01" and row.test_date<="2020-05-31":
        x_train_res.at[i,"test_date"]= 1
    elif row.test_date>="2020-10-01" and row.test_date<="2020-12-31":
        x_train_res.at[i,"test_date"]= 1
    elif row.test_date>="2021-01-01" and row.test_date<="2021-05-31":
        x_train_res.at[i,"test_date"]= 1
    elif row.test_date>="2021-10-01" and row.test_date<="2021-12-31":
        x_train_res.at[i,"test_date"]= 1
    elif row.test_date>="2020-06-01" and row.test_date<="2020-09-30":
        x_train_res.at[i,"test_date"]= 0
    elif row.test_date>="2021-06-01" and row.test_date<="2021-09-30":
        x_train_res.at[i,"test_date"]= 0
    i= i+1

x_test= x_test.reset_index()
y_test= y_test.reset_index()
i=0
#normalizziamo le date del test set
for row in x_test.iterrows():
    if row.test_date>="2020-01-01" and row.test_date<="2020-05-31":
        x_test.at[i,"test_date"]= 1
    elif row.test_date>="2020-10-01" and row.test_date<="2020-12-31":
        x_test.at[i,"test_date"]= 1
    elif row.test_date>="2021-01-01" and row.test_date<="2021-05-31":
        x_test.at[i,"test_date"]= 1
    elif row.test_date>="2021-10-01" and row.test_date<="2021-12-31":
        x_test.at[i,"test_date"]= 1
    elif row.test_date>="2020-06-01" and row.test_date<="2020-09-30":
        x_test.at[i,"test_date"]= 0
    elif row.test_date>="2021-06-01" and row.test_date<="2021-09-30":
        x_test.at[i,"test_date"]= 0
    i= i+1

#riportiamo i dataset agli indici originali altrimenti avremo il campo index
x_train_res= x_train_res.set_index("index")
y_train_res= y_train_res.set_index("index")
x_test= x_test.set_index("index")
y_test= y_test.set_index("index")

print("Fine Feature Scaling!")

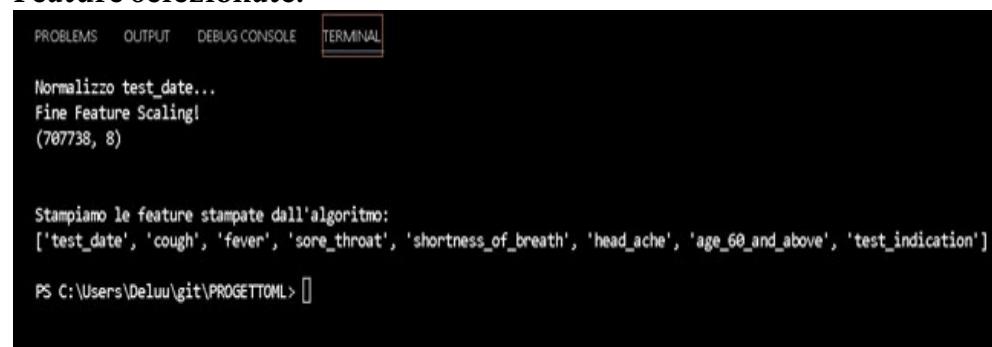
```

3.5 Feature Engineering:

Sulla base dei dati ottenuti fin ora, abbiamo analizzato le feature e attraverso la conoscenza del dominio siamo arrivati alla conclusione che le 9 variabili indipendenti sono molto rilevanti per la predizione. Nonostante ciò, andremo comunque ad effettuare feature selection per confrontare i risultati che otterremo dal modello di machine learning (analizziamo il modello applicando feature selection e non). Per i dati su cui attuiamo feature selection utilizzeremo la tecnica dell'eliminazione univariata di feature attraverso l'algoritmo SelectKbest, quindi andremo a selezionare le K feature migliori correlate con la variabile dipendente. In questo caso il K scelto è 8.

```
#inizio feature selection
fs = SelectKBest(score_func=chi2,k=8)
fs.fit_transform(x_train_res, y_train_res)
#otteniamo il dataset solo delle feature selezionate
x_new_train_res = fs.transform(x_train_res)
#eliminiamo anche queste feature dal test set.NB la feature selection è stata applicata solo sul training set
x_new_test = fs.transform(x_test)
print(x_new_train_res.shape)restituisce il numero di righe
#printiamo le feature stampate dall'algoritmo
print("\n\nStampiamo le feature stampate dall'algoritmo:")
x.columns[fs.get_support(indices=True)]
print(x.columns[fs.get_support(indices=True)].tolist())
```

Feature selezionate:



The screenshot shows a Jupyter Notebook terminal window. At the top, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL selected. The terminal output displays the following text:

```
Normalizzo test_date...
Fine Feature Scaling!
(707738, 8)

Stampiamo le feature stampate dall'algoritmo:
['test_date', 'cough', 'fever', 'sore_throat', 'shortness_of_breath', 'head_ache', 'age_60_and_above', 'test_indication']

PS C:\Users\Deluu\git\PROGETTOML> []
```

4. Data Modeling:

Dato che ci troviamo di fronte ad un problema di classificazione, ovvero ad un'istanza di un problema di apprendimento supervisionato; abbiamo deciso di utilizzare due tecniche differenti per realizzare il modello:

- Decision Tree (algoritmo basato su entropia).
- Naive Bayes (algoritmo probabilistico).

In questo modo, nella fase successiva di validazione, analizzeremo i risultati ottenuti dei due modelli in modo da poterli confrontare e determinare il modello migliore da utilizzare per il problema in questione. (Ovviamente li setteremo anche su diverse configurazioni delle fasi precedenti).

- Report dei risultati del Decision Tree con data balancing e senza feature selection:

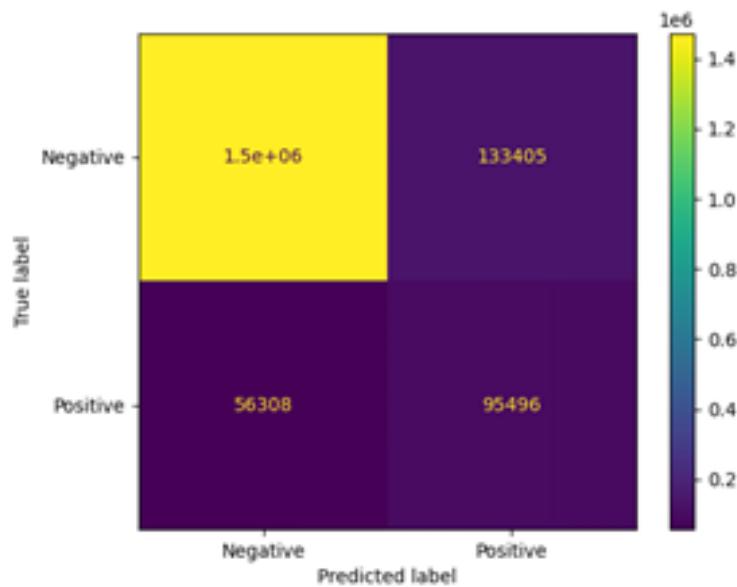
```

[[1473235 133405]
 [ 56308  95496]]
      precision    recall   f1-score   support
Negative       0.96       0.92       0.94     1606640
Positive        0.42       0.63       0.50     151804

accuracy          0.89
macro avg        0.69       0.77       0.72     1758444
weighted avg     0.92       0.89       0.90     1758444

Accuracy: 0.8921131409359638

```

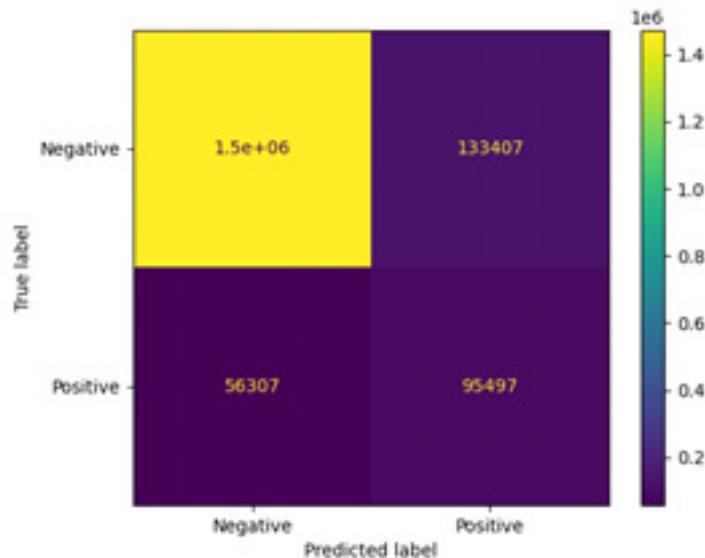


- Report dei risultati del Decision Tree con data balancing e con feature selection:

```
[[1473233  133407]
 [ 56307   95497]]
      precision    recall   f1-score   support
Negative       0.96     0.92     0.94    1606640
Positive        0.42     0.63     0.50    151804

accuracy          0.89
macro avg       0.69     0.77     0.72    1758444
weighted avg     0.92     0.89     0.90    1758444

Accuracy: 0.8921125722513767
```

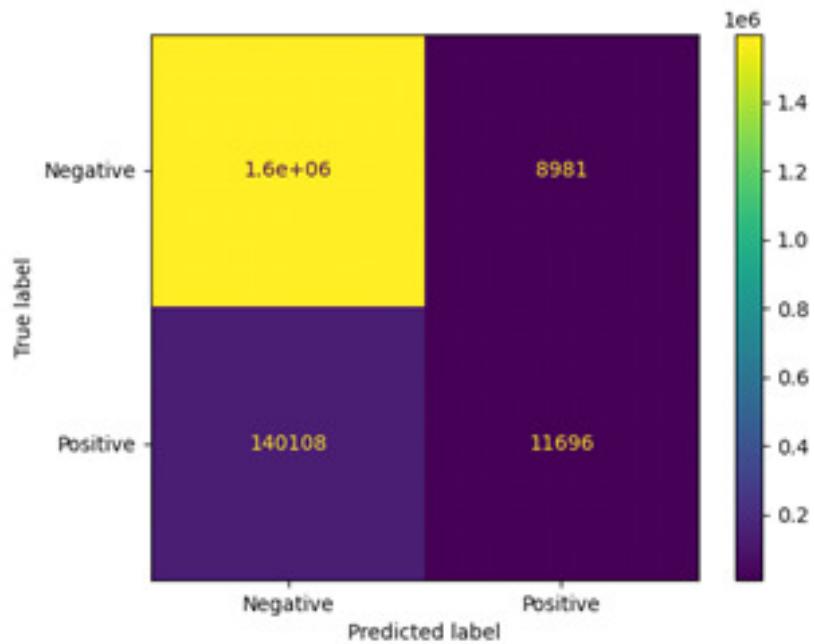


- Report dei risultati del Decision Tree senza applicare data balancing e feature selection:

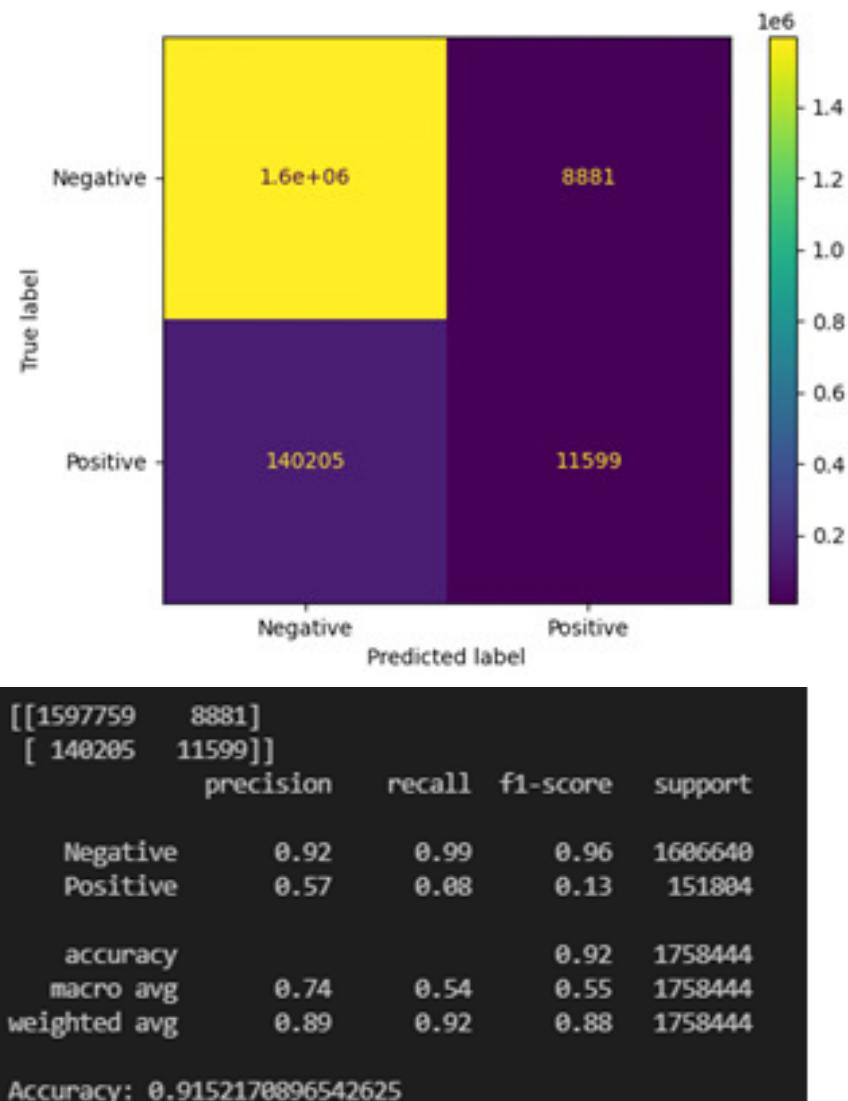
```
[[1597659    8981]
 [ 140108   11696]]
      precision    recall   f1-score   support
Negative      0.92      0.99      0.96   1606640
Positive       0.57      0.08      0.14   151804

accuracy          0.92      0.92      0.92   1758444
macro avg       0.74      0.54      0.55   1758444
weighted avg     0.89      0.92      0.88   1758444

Accuracy: 0.9152153836005014
```



- Report dei risultati del Decision Tree senza applicare data balancing ma applicando feature selection:

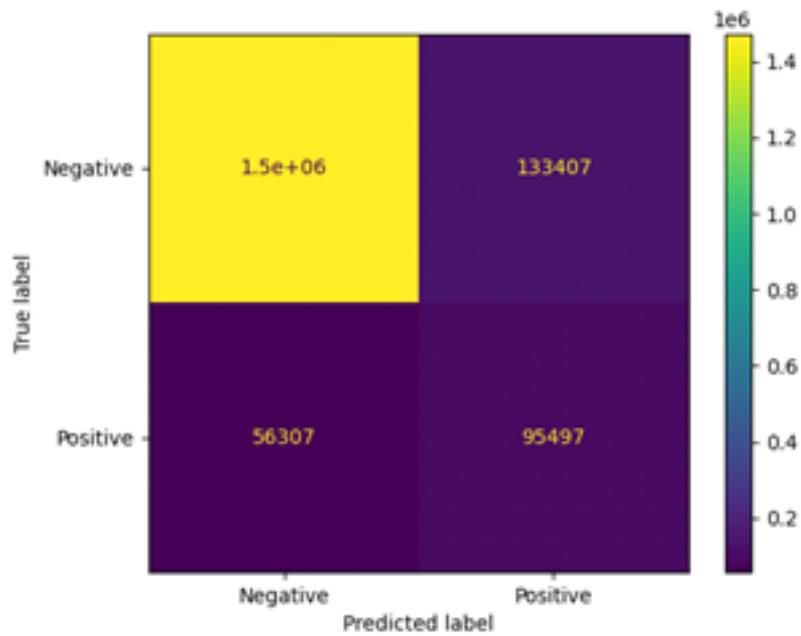


- Report dei risultati del Naive Bayes con data balancing e con feature selection:

```
[[1473233 133407]
 [ 56307  95497]]
      precision    recall   f1-score   support
Negative       0.96     0.92     0.94    1606640
Positive       0.42     0.63     0.50    151804

accuracy          -         -     0.89    1758444
macro avg       0.69     0.77     0.72    1758444
weighted avg     0.92     0.89     0.90    1758444

Accuracy: 0.8921125722513767
```

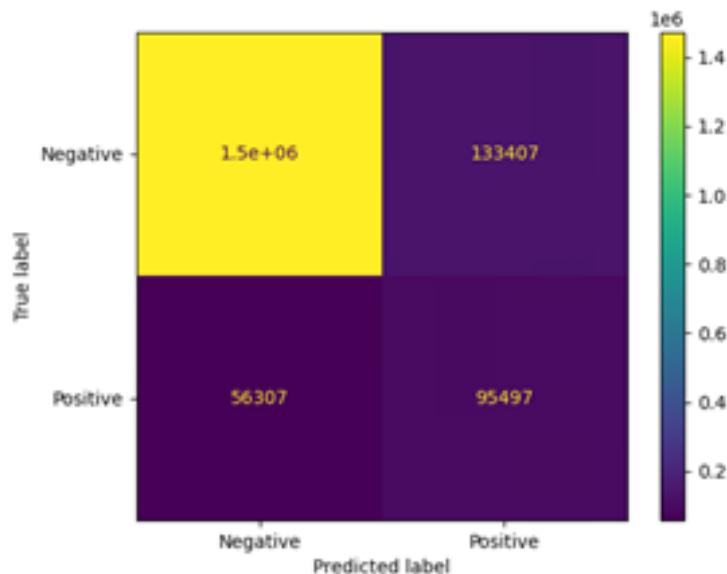


- Report dei risultati del Naive Bayes con data balancing e senza feature selection:

```
[[1473233 133407]
 [ 56307  95497]]
      precision    recall   f1-score   support
Negative       0.96      0.92      0.94     1606640
Positive       0.42      0.63      0.50     151804

accuracy          0.89
macro avg       0.69      0.77      0.72     1758444
weighted avg    0.92      0.89      0.90     1758444

Accuracy: 0.8921125722513767
```



- Report dei risultati del Naive Bayes senza applicare data balancing ma applicando feature selection:

```

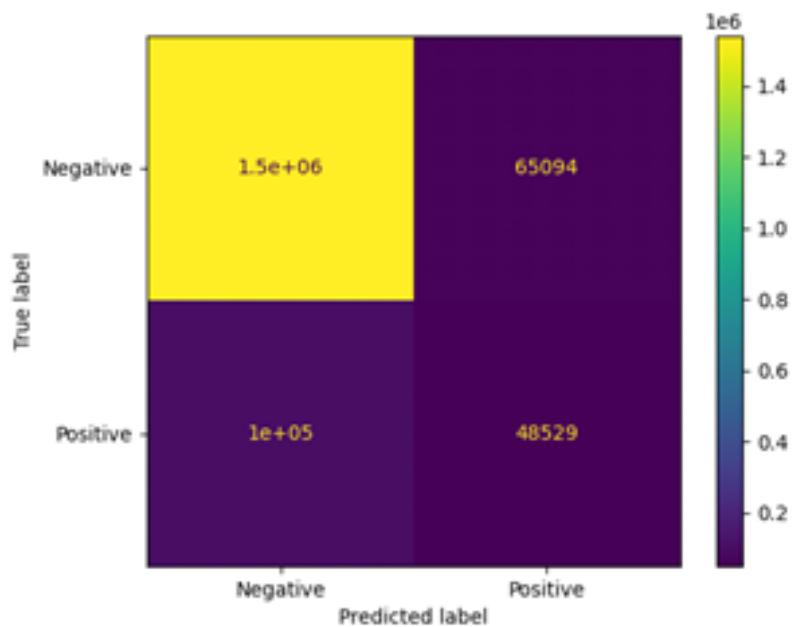
[[1541546  65094]
 [ 103275  48529]]
      precision    recall   f1-score   support
Negative       0.94      0.96      0.95     1606640
Positive       0.43      0.32      0.37     151884

accuracy          -         -      0.90     1758444
macro avg       0.68      0.64      0.66     1758444
weighted avg     0.89      0.90      0.90     1758444

Accuracy: 0.9042511447628737

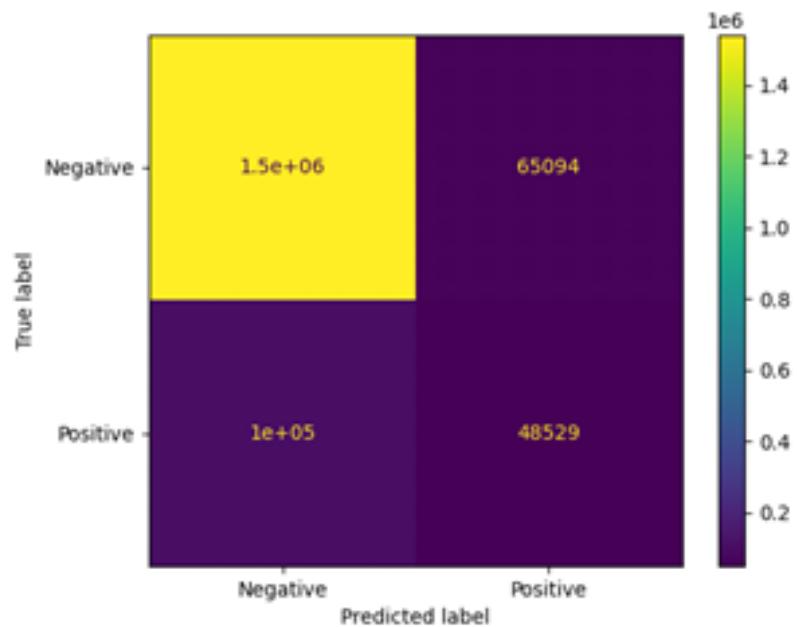
```

ù



- Report dei risultati del Naive Bayes senza applicare data balancing e feature selection:

[[1541546 65094]	
[103275 48529]]	
precision recall f1-score support	
Negative	0.94 0.96 0.95 1606640
Positive	0.43 0.32 0.37 151884
accuracy	0.90 1758444
macro avg	0.68 0.64 0.66 1758444
weighted avg	0.89 0.90 0.90 1758444
Accuracy: 0.9042511447620737	



5. Evaluation

Osservando i risultati ottenuti applicando i due classificatori, ovvero Naive Bayes e Decision Tree, notiamo che quest'ultimo ha un'accuracy leggermente migliore di Naive Bayes, mentre per le altre metriche otteniamo gli stessi esiti. È comunque da tener in considerazione che il nostro progetto tratta un problema di natura sbilanciato e anche applicando il balancing, il classificatore sarà comunque più bravo a stabilire se una persona è negativa rispetto a stabilire se una persona è positiva. Quindi non possiamo fidarci ciecamente delle metriche, infatti nel visualizzare i risultati ottenuti senza applicare data balancing, potremmo essere tratti in inganno, dato che la precision per i positivi è aumentata così come per l'accuracy; questo perché il classificatore ha più istanze negative da cui poter apprendere e di conseguenza è più bravo a differenziare quest'ultimi dai positivi e quindi ottiene un'accuracy e una precisione migliore (ma possiamo notare che la recall dei positivi è diminuita drasticamente). Possiamo affermare che il data balancing deve essere applicato altrimenti non possiamo fidarci delle metriche; ma per quanto riguarda l'operazione di feature selection, notiamo che il cambiamento degli esiti nell'applicare o meno questa tecnica non è significante poiché è minimo. Infatti, la feature eliminata (gender) non è molto rilevante per la predizione che deve effettuare il modello. Ovviamente se vogliamo avere una maggiore velocità di apprendimento del modello dobbiamo applicare la tecnica di feature selection anche se la differenza è lieve. Quindi per il problema in analisi conviene applicare come classificatore il Decision Tree con la seguente configurazione: applicando sia data balancing e sia feature selection. Inoltre il nostro progetto rispetta i business success criteria in quanto bisogna avere almeno il 50% di accuracy ma non viene richiesto che il modello abbia una certa velocità e quindi potremmo scegliere una delle due configurazioni (applicando feature selection o non).