

Università degli Studi di Salerno

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Data Science e Machine Learning

**Re-Identification**

De- anonimizzazione di un dataset tramite algoritmi di machine learning

**Studenti**

Annunziata Gianluca **Mat.**0522500723

De Rosa Gerardo **Mat.**0522500722

**Docente Prof.** Giuseppe Polese

Anno Accademico 2018-2019

# Sommario

[Sommario 2](#_Toc29576118)

[Introduzione 3](#_Toc29576119)

[1. Lo stato dell’arte 4](#_Toc29576120)

[1.1 Tecniche di De-anonimizzazione 4](#_Toc29576121)

[1.2 “Finding a Needle in a Haystack” [3] 4](#_Toc29576122)

[1.3 “De-Anonymizing Social Networks with Random Forest Classifier” [4] 5](#_Toc29576123)

[1.5 “A Structure-based De-anonymization Attack on Graph Data Using Weighted Neighbor Match” [5] 5](#_Toc29576124)

[2. Sviluppo dell’applicazione 6](#_Toc29576125)

[2.1 L’algoritmo 8](#_Toc29576126)

[2.2 Il testing 11](#_Toc29576127)

[2.2.1 Roc Curve 11](#_Toc29576128)

[2.2.2 Confusion Matrix 12](#_Toc29576129)

[2.3 Features Selection & AdaBoost 16](#_Toc29576130)

[2.3.1 Features Selection 16](#_Toc29576131)

[2.3.2 AdaBoost 17](#_Toc29576132)

[3. Sviluppi futuri 18](#_Toc29576133)

[4. Conclusioni 18](#_Toc29576134)

[Riferimenti 20](#_Toc29576135)

# Introduzione

Nel corso di questo paper descriveremo il nostro approccio al problema della re-identificazione, ovvero alla pratica di ricercare l’identità di una o più persone a cui si riferiscono dati anonimizzati presenti in uno o più dataset, tramite algoritmi di machine learning. Risulta interessante affrontare questa problematica poiché moltissime compagnie con privacy policies, in campi che spaziano da quello finanziario a quello della salute, rilasciano al pubblico dati dopo che questi hanno subito un processo di de-identificazione, ovvero la rimozione dagli stessi di alcuni di matrice personale, come generalità, indirizzi o data di nascita; ciò avviene di solito tramite processi di masking, generalizzazione o con appunto la cancellazione degli stessi. I dati che subiscono questa trasformazione si dividono in identificatori diretti ed indiretti ma il grado tramite il quale avviene questo processo di de-identificazione è differente per ogni dataset (grandi insieme di dati) e non sempre vengono quindi eliminate o camuffate entrambe le tipologie, molto spesso anzi il processo che viene applicato su quelli di tipo diretto li trasforma in identificatori di tipo indiretto.

Risulta quindi necessario utilizzare i dati disponibili in combinazione con tecniche di data science ed in particolari con metodologie di machine learning per approntare un riconoscimento dei dati anonimi che sia quanto il più efficace e vicino alla realtà possibile; tutto ciò è aiutato dal fatto che con l’avvento dei big data e la loro diffusione, moltissimi dataset di questa tipologia sono reperibili in rete, permettendo con il loro utilizzo di affinare sempre più gli algoritmi che operano in questo senso.

Nel nostro caso di seguito descriveremo di come utilizzando un dataset prelevato dal sito UCI machine learning, abbiamo operato attraverso l’utilizzo di algoritmi di ***Random Forest*** [1]e di identità create ad-hoc una re-identificazione sui dati, sfruttando le diverse feature (dati per ogni individuo) presenti nel dataset, individuandone inoltre le più decisive nel riconoscimento. Prima però di partire con la descrizione dello sviluppo del nostro progetto, analizzeremo lo stato dell’arte nel campo della ***Re-Identification*** [2]***,*** concludendo poi il tutto con i nostri pareri sui possibili sviluppi futuri sia per quanto riguarda la problematica in generale che per il nostro algoritmo in particolare.

# 1. Lo stato dell’arte

Di seguito verranno descritte alcune delle metodologie di re-Identification più popolari nel mondo della data science negli ultimi anni; Verranno citati alcuni paper che abbiamo consultato durante il nostro lavoro, in particolare saranno descritte le idee alla base dei loro algoritmi, oltre allo scopo in dettaglio per cui sono stati pensati.

## Tecniche di De-anonimizzazione

La re-identificazione è impiegata per lo più nel contesto dei social network, e i ¾ degli studi reperibili in letteratura per essere analizzati riguardano questa categoria.

Uno dei metodi tradizionali per effettuare il riconoscimento è il data matching, altre tecniche minormente utilizzate consistono nell’utilizzo di nodi, archi oppure nell’uso di grafi, allo scopo di identificare soggetti o account, tramite matching delle informazioni che vi sono contenute.

## 1.2 “Finding a Needle in a Haystack” [3]

La problematica studiata in questo paper concernente la re-identificazione, ha a che fare con l’identificare, se esistono, i record unici presenti ne record anonimi Census (dati provenienti da censimenti) e provare a mascherare questi dati in modo da proteggerli da eventuali attacchi di re-identificazione. Per fare in primis viene scelto un dataset adatto che sia nell’ordine del 10 000 000, che abbia almeno 20 o più variabili e che per almeno una o più di queste il numero di categorie debba superare le 10; poi vengono testate due diverse tecniche per identificare i record unici, basate principalmente sull’ordinamento e su diverse tipologie di trasformazioni di dati studiate ad-hoc; ed infine sono presentate alcune idee per quanto riguarda la tutela dei dati, una volta identificati quelli sensibili. A questo scopo sono prese in considerazione 5 alternative: distruggere i record contenti i dati unici rilevati; rendere vuoti i campi dati unici rilevati; utilizzare qualche tecnica di perturbazione dei dati o infine criptare i dati.

Da questo paper abbiamo appreso l’importanza di badare ai dati unici presenti nel dataset e soprattutto a come questi per ogni categoria contribuiscano a creare l’unicità dei record, poiché sfruttando questa proprietà è possibile aumentare in maniera consistente l’accuratezza dell’associazione di un’identità ad un record piuttosto che ad un altro.

## 1.3 “De-Anonymizing Social Networks with Random Forest Classifier” [4]

Il paper in questione ha come scopo quello di provare la possibilità di identificare le persone che utilizzano i social network dai dati anonimizzati da questi resi pubblici. Per fare ciò vengono prelevati dati anonimi da social network e convertiti in modo da trasformare il problema da uno di de-anonimizzazione ad uno di classificazione binaria tra coppie di nodi. Vengono poi individuate le features ricavate dalla struttura della rete ottenuta per allenare un classificatore Random Forest, che come risultato effettua un matching tra le coppie nodi candidate (reti anonime) e le reti ausiliari.

Pur discostandosi molto dalla nostra problematica, questo paper ci ha forniti spunti interessanti per quanto riguarda la tecnologia da utilizzare, la pulizia dei dati e l’inserimento di rumore per migliorare la robustezza dell’algoritmo.

## 1.5 “A Structure-based De-anonymization Attack on Graph Data Using Weighted Neighbor Match” [5]

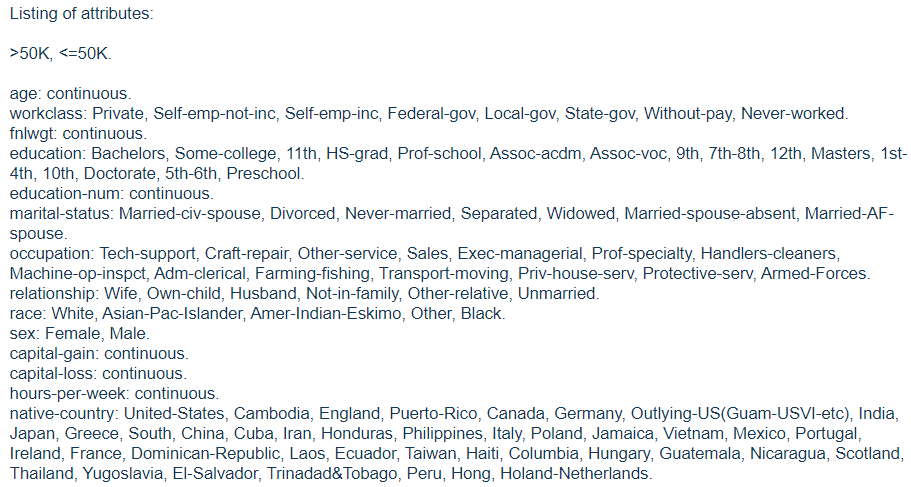
Lo studio effettuato in questo paper affronta la tematica della de-anonimizzazione, impiegando una tecnica chiamata *“weighted neighborhood matching algorithm* ***(SWNM)****”* che basa la sua efficacia sulla considerazione complessiva di features globali e locali quando calcola la similarità tra nodi anonimizzati ed ausiliari.

Lo studio nasce sia come risposta a tutti i metodi di de-anonimizzazione che si basano su seed, poiché afferma che questo metodo non è in grado di generalizzare quando si tratta di identificare, nonostante alcune match che identifica siano evidentemente corretti; sia come risposta alla tipologia di re-identificazione che senza seed, cerca di effettuare match prevalentemente considerando le informazioni sulle strutture del network, poiché questa necessità di un features per nodo ben definite e una formula capace di calcolare la similarità molto accurata.

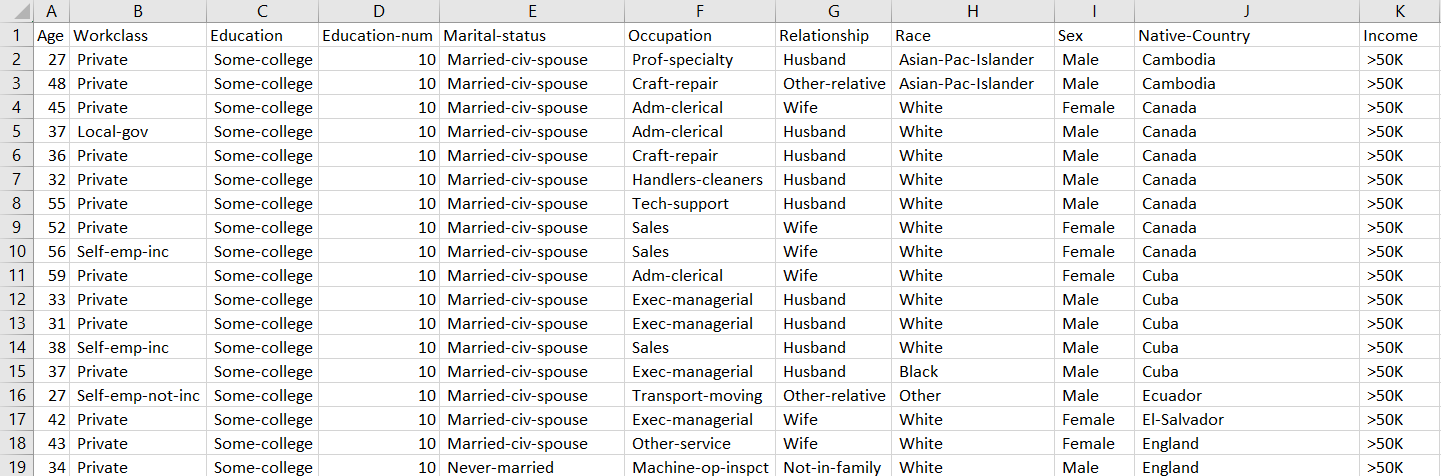
Il metodo che quindi viene proposto in questo paper utilizzando le informazioni ottenute dal metodo “weighted neighborhood” e una matrice di similarità dinamica procede a calcolare la similarità tra grafi anonimi ed ausiliari, impiegando come detto in precedenza features locali e globali. Tra le conclusioni viene evinto come l’algoritmo si comporti bene con i dataset con i quali viene testato, ma che è poco incline alla adattabilità a differenti “rumori” nei dati; ciò ci ha fatto riflettere che anche nel nostro caso pur ottenendo risultati molto buoni, fosse necessario aggiungere del rumore, per testare proprio l’adattabilità dell’algoritmo e produrre un predittore maggiormente efficiente.

# 2. Sviluppo dell’applicazione

Per lavorare sulla problematica oggetto del nostro studio, ovvero la re-identificazione, come primo passo abbiamo scaricato dal sito ***UCI – machine learning*** [6], un dataset chiamato ***“adult”,*** contenente dati anonimizzati utilizzati in un algoritmo di machine learning atto a classificare i soggetti in due categorie, a seconda che il loro reddito fosse minore o superiore ai 50K annui; gli attributi o per meglio dire features per ogni soggetto risultano essere:

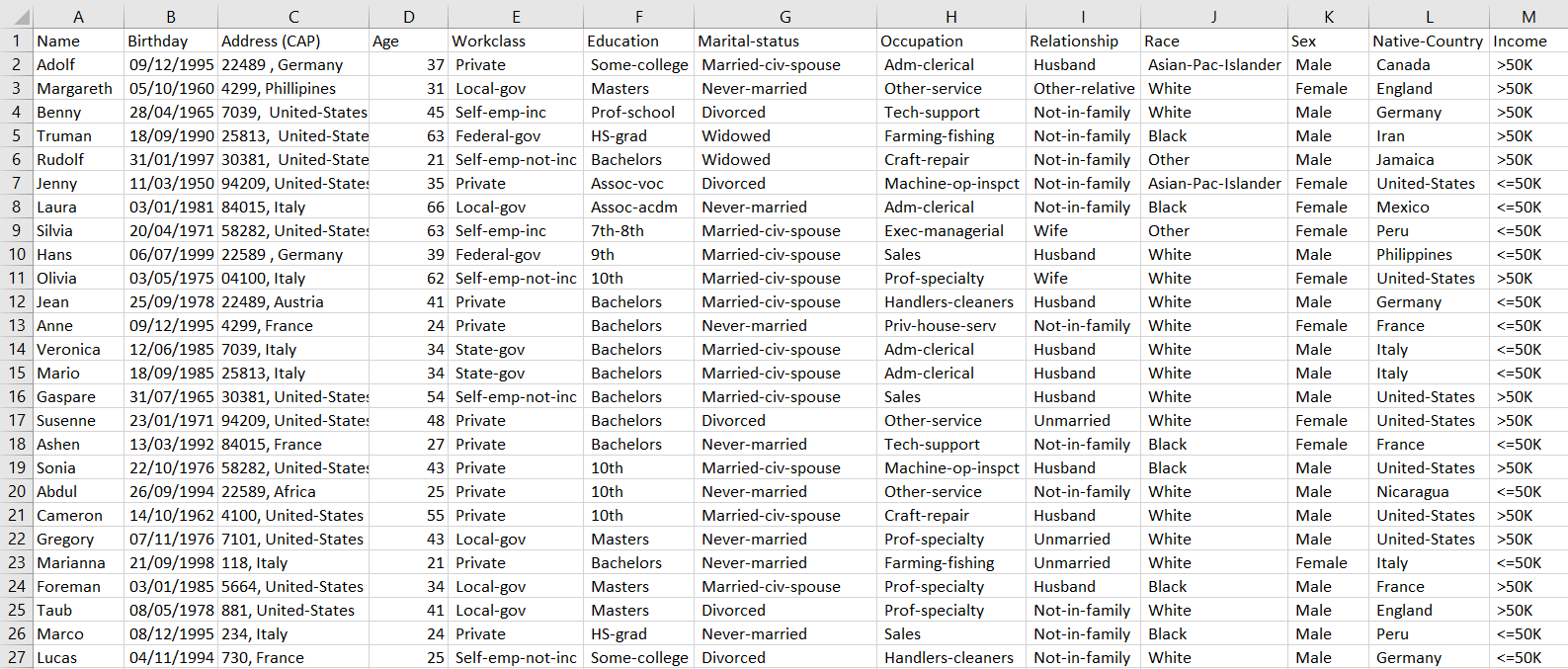


Il dataset composto quindi da ***14 attributi e 32561 soggetti***, è stato da noi sottoposta ad una **pulizia**; prima di tutto si è proceduto ad eliminare le righe nulle, e quelle contenenti dati non congrui; poi sono state eliminate anche alcune features ritenute non fondamentali per il record linking, di cui parleremo in seguito; dopo tutte le trasformazioni, il dataset è arrivato a constare di ***11 attributi e 30163 soggetti;*** di seguito una piccola porzione del dataset:



Come si può notare sono stati eliminati solo gli attributi che presentavano valori nel campo dei numeri continui, poiché appunto non sarebbero stati rilevanti nel ***record linkage*** [7]***,*** non essendo fondamentali e anzi potenzialmente dannosi nel differenziare un individuo da un altro.

Una volta preparato il dataset abbiamo creato il dataset che avremmo utilizzato poi per effettuare la re-identificazione, ovvero un dataset contenente gli stessi attributi del dataset ***adult*** ma con in più dati sensibili, come nome cognome, indirizzo e data di nascita, opportunamente inventati ad-hoc per rispecchiare i soggetti anonimizzati ai quali si riferivano:

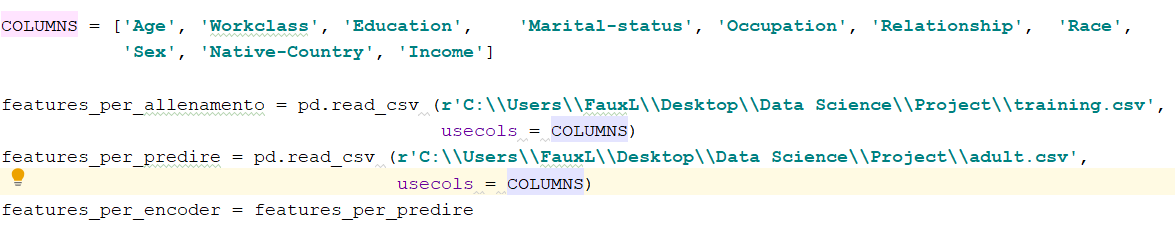


Dopo aver quindi preparato i dataset, il nostro compito è stato quello di trovare una metodologia nel campo della **data science** che potesse essere utilizzata per implementare via codice la nostra idea; dopo alcune ricerche, la scelta è quindi ricaduta sull’utilizzo del ***random forest*** come metodologia d’apprendimento***, python*** come linguaggio e ***pycharm*** come ambiente di sviluppo.

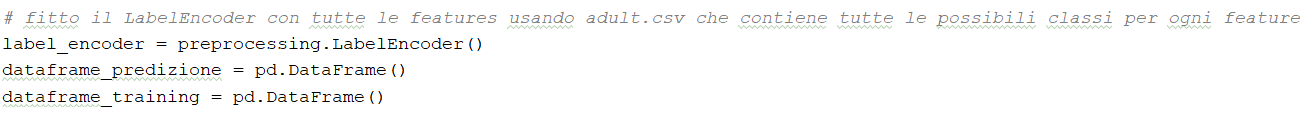
Decisi l’ambiente, la tecnologia e le tecniche da impiegare siamo quindi passati all’azione e alla formulazione di un’idea che ci permettesse di istanziare uno script efficiente per la riuscita dello scopo che ci eravamo prefissati, ovvero il creare un algoritmo che associasse un’identità ai soggetti presenti nel dataset in base all’identità conosciuta, che più gli si avvicinasse presente nel dataset da noi creato.

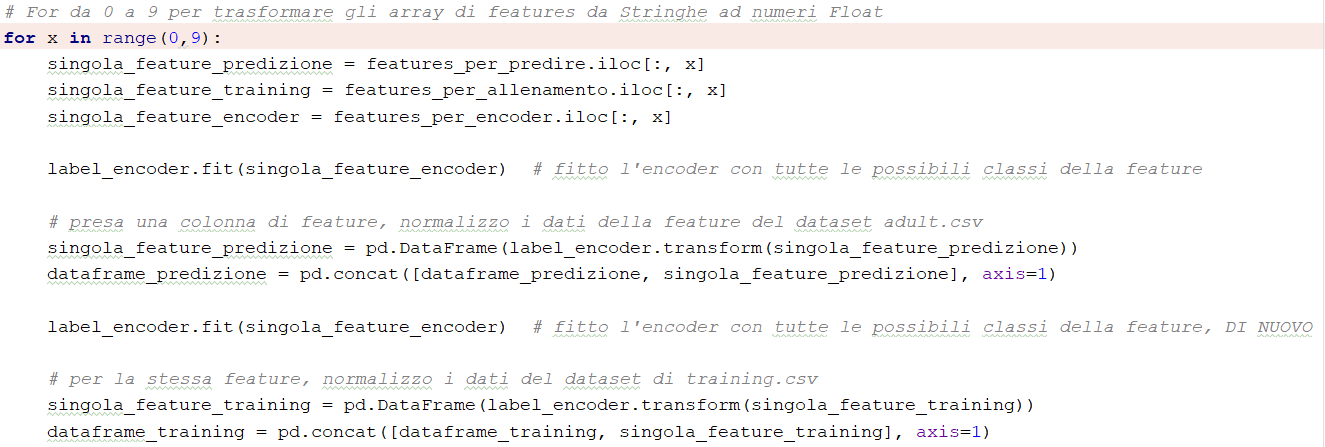
## 2.1 L’algoritmo

Come primo passo, dopo aver importate alcune librerie, abbiamo proceduto ad importare i due dataset all’interno di ***pycharm***, tramite il seguente codice:

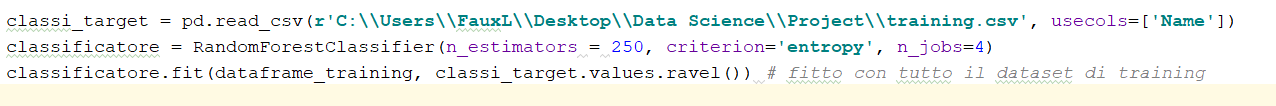


Dopo aver quindi prelevato i dataset è entrata in gioco la problematica di come gestire questi dati, poiché l’algoritmo random forest, non lavora con le stringhe, ed i nostri dataset erano formati in gran parte da esse, abbiamo proceduto ad effettuare un **encoding** dei dati, sia del dataset adult che di quello da noi istanziato; per far si però che i dati dopo aver effettuato l’encoding rimanessero comunque congrui e corrispondenti nei due dataset, abbiamo allenato prima l’encoder sul dataset adult e poi applicato su entrambi:

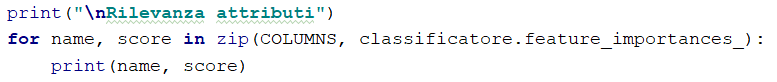


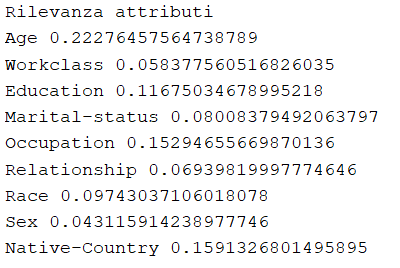


Dopo aver quindi ottenuto dataset completamente numerici procediamo a creare un **classificatore**, con **250 estimatori**, basato sull’**entropia**, e ad importare le **label** per poter effettuare il **fitting** [8]***:***



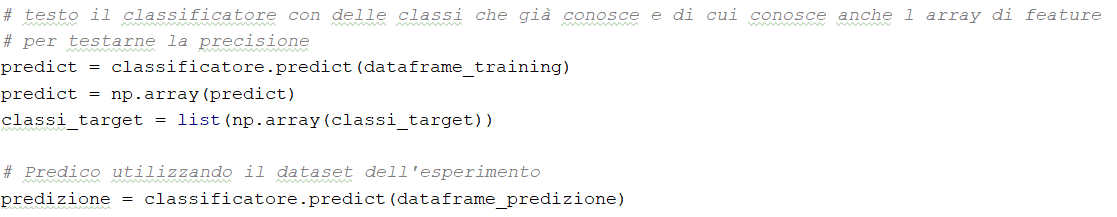
Fatto ciò cominciamo ad ottenere i primi dati utili da quanto realizzato, in particolare si tratta della rilevanza delle features nel riconoscimento di ognuna, di seguito mostrata:





Come si nota dai risultati, gli attributi che risultano essere più rilevanti, sono: ***age, education, occupation, native country*** e infine ***race;*** ciò non ci ha sorpresi poiché questi attributi sono quelli che hanno più varianza tra tutti; ovvero più righe con valori diversi tra di loro.

Con il codice seguente abbiamo proceduto a testare sul data frame di training per ottenere un’idea della precisione; per poi testare sull’intero data frame:

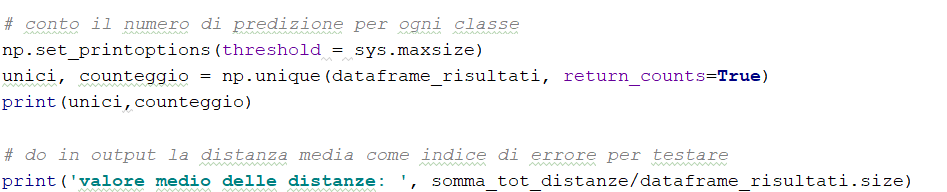


Dopo aver effettuato la predizione decidiamo di calcolare un indice sulla vicinanza di ciascun soggetto alla “categoria” a cui viene predetto appartenere, pur non essendo valutabile come “test” sulla bontà dell’applicativo, utilizzando quindi la ***distanza euclidea*** [9]abbiamo ottenuto un’informazione in più riguardo al comportamento del classificatore; il codice tramite il quale abbiamo effettuato il calcolo delle distanze è il seguente:

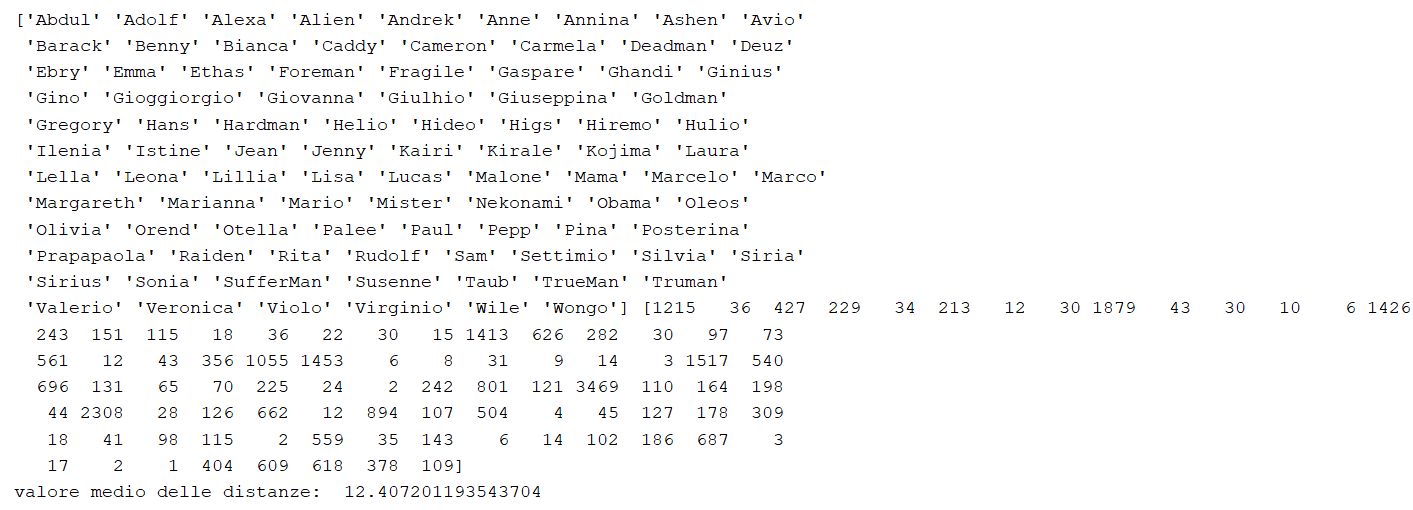


Dopo aver quindi predetto e calcolato le distanze, abbiamo inserito questi dati in due file Excel, uno per ciascuna delle due misure calcolate.

Come altro indice, abbiamo poi pensato di sommare le volte in cui ciascuna identità venisse assegnate nel dataset di test e di stampare in output il vettore corrispondente; infine abbiamo pensato di stampare anche la distanza euclidea media; cosi da aver sempre disponibile un’indicazione sulla lontananza media dalla categoria a cui ciascun soggetto viene assegnato:



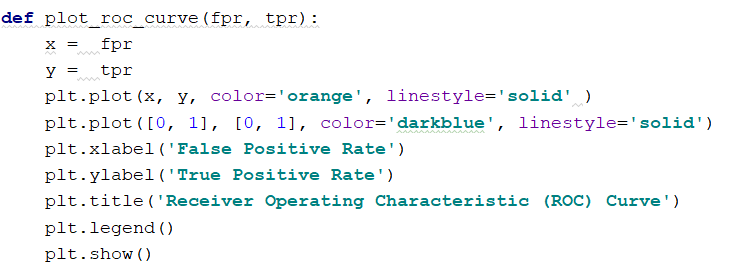
Poiché con valori presi direttamente dal dataset sarebbe stato facile ottenere solo dati molto diversi fra di loro, abbiamo deciso di aggiungere del rumore ai dati, che ha sì abbassato le prestazioni del nostro algoritmo, ma ha impedito l’adattamento dell’algoritmo di classificazione permettendo allo stesso di lavorare meglio sui dati; alla luce di ciò, l’output del codice di cui sopra risulta essere:



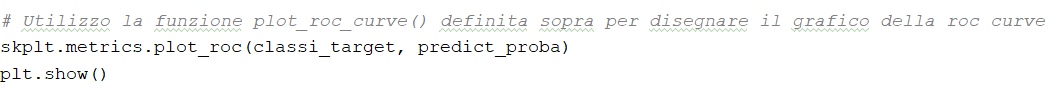
## 2.2 Il testing

Dopo aver lavorato quindi alla creazione dell’algoritmo, siamo passati alla fase di testing, ovvero al testing dell’accuratezza dello stesso; per ottemperare a tale scopo abbiamo preso in esame due metodologie differenti: ***Roc Curve*** [10]e la ***Confusion Matrix*** [11]***.***

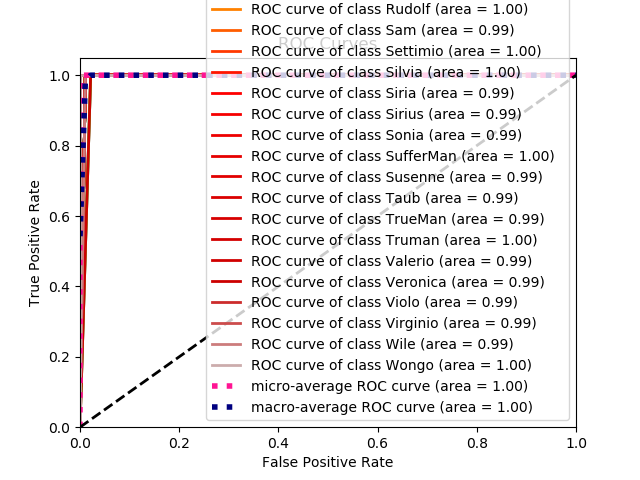
### 2.2.1 Roc Curve

Come primo passo abbiamo definito una funzione per la creazione e il plotting della Roc Curve: 

Fatto ciò con il seguente codice abbiamo provveduto al plot della stessa:

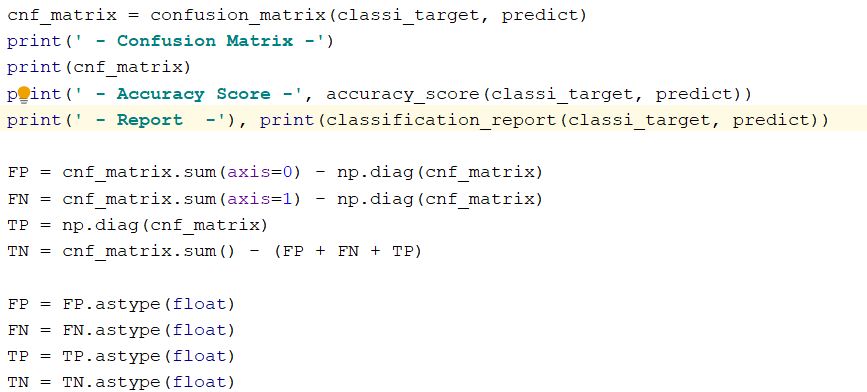


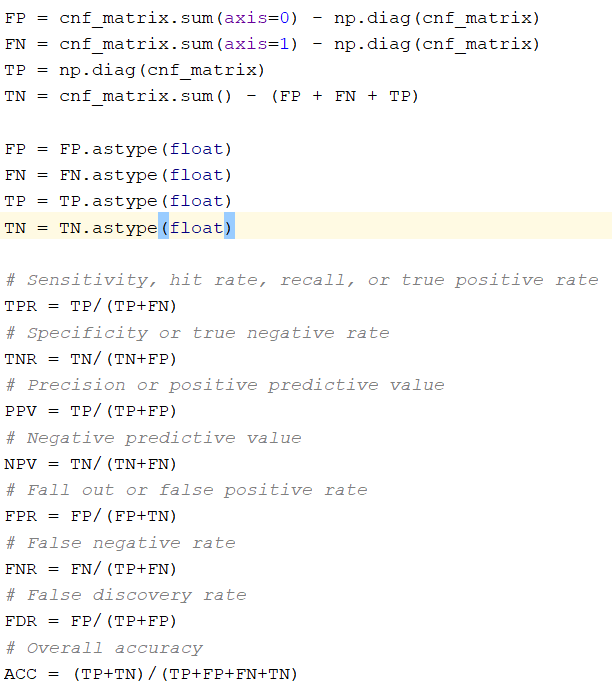
La curva ottenuta, mostrata in pagina seguente, mostra come per la maggior parte delle categorie avvenga un’assegnazione vicina all’uno, con una grande preponderanza di true positive e qualche sporadico falso positivo; il risultato è più che accettabile considerando come abbiamo già detto in precedenza che all’interno dei dati è stato inserito del rumore, proprio per rendere più “reale” il campione e difficile la predizione.



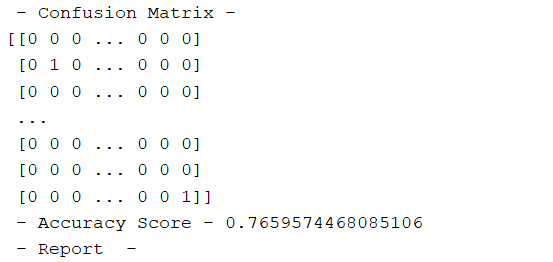
### 2.2.2 Confusion Matrix

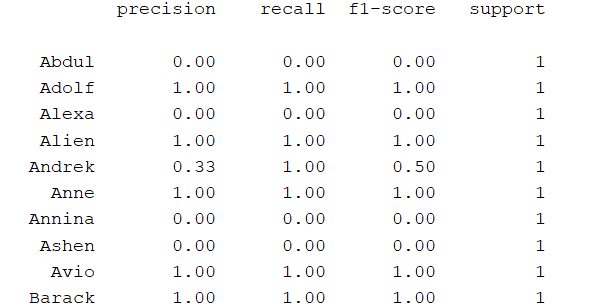
Per quanto riguarda la confusion matrix di seguito è illustrato il codice tramite il quale quest’ultima è stata calcolata:

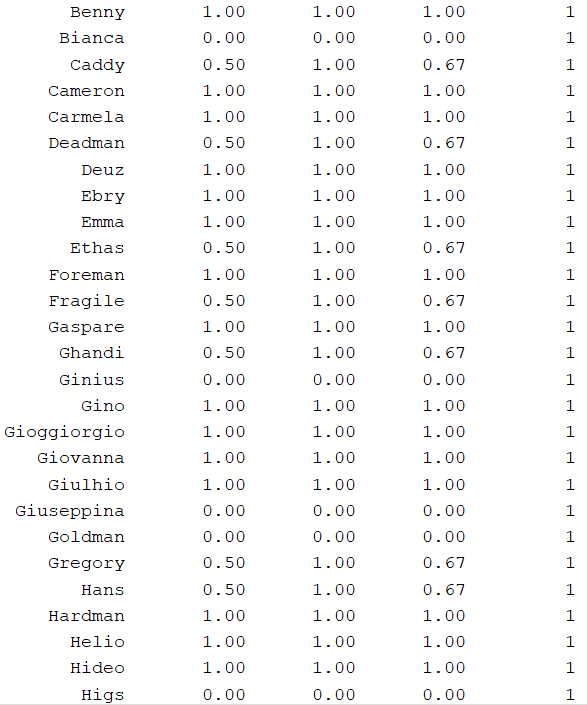
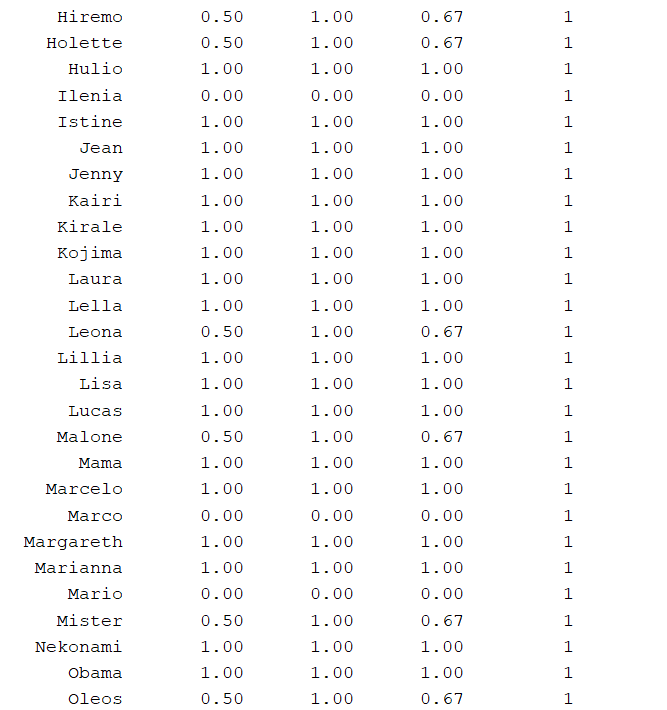


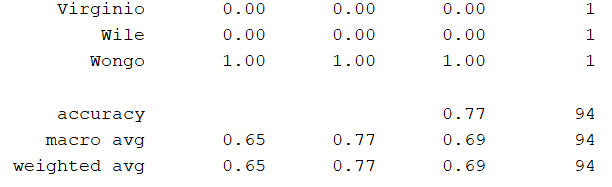
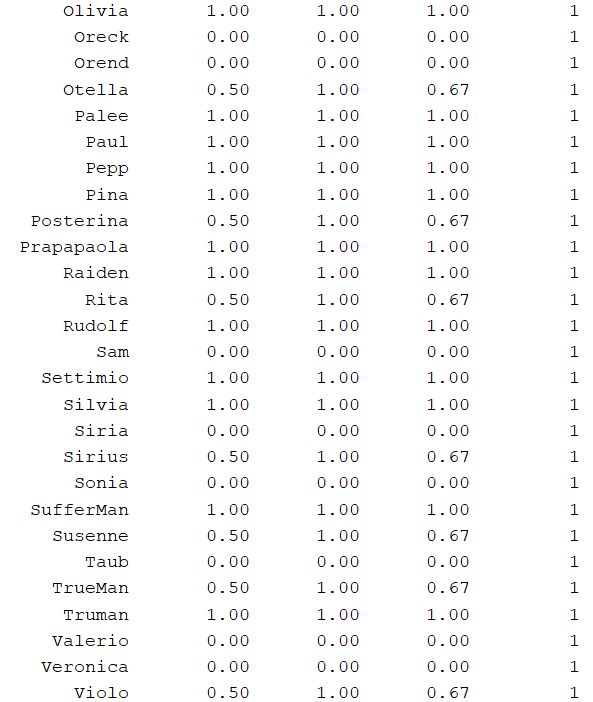


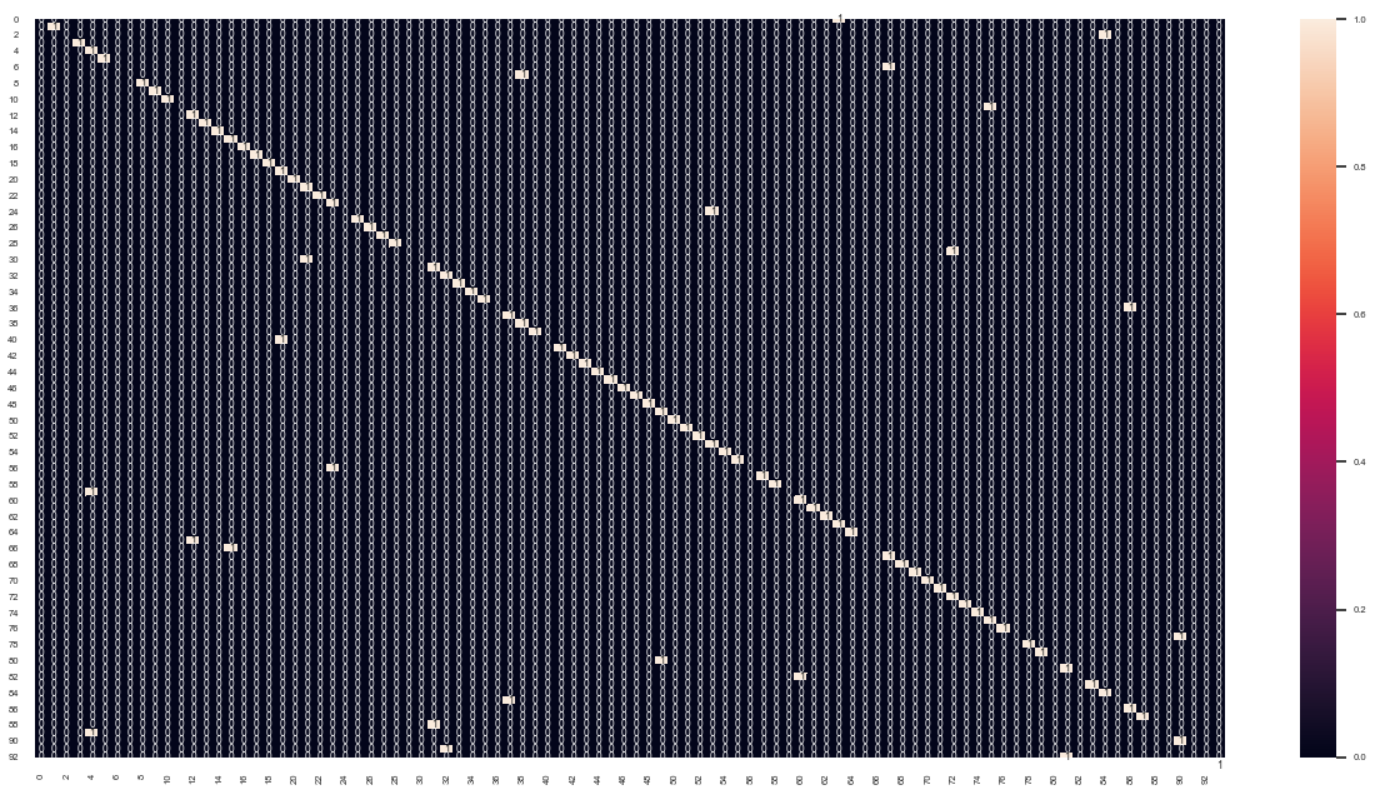
Otteniamo il seguente output:



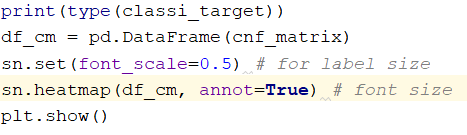








Quella sopra riportata è la confusion matrix, se tutte le label fossero correttamente assegnate, la diagonale dovrebbe essere composta soltanto da 1, invece nel nostro caso, il classificatore etichetta in modo errato in diversi casi; per questo motivo, la percentuale di accuratezza restituita dalla C.M. è ***0.77;*** *il* codice che abbiamo utilizzato per effettuare il plotting è quello seguente:



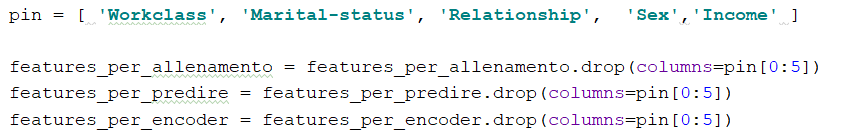
## 2.3 Features Selection & AdaBoost

Dopo aver completato la costruzione dell’algoritmo ed aver ultimato i test, abbiamo provato a migliorare i risultati da noi ottenuti, e ciò è stato fatto secondo due tecniche: l’utilizzo per la classificazione delle sole features più incidenti (operare quindi una ***features selection***); e l’utilizzo dell’algoritmo ***AdaBoost.***

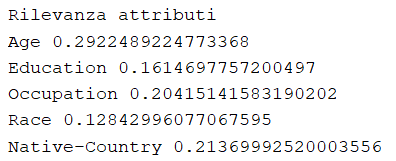
### 2.3.1 Features Selection

Come abbiamo mostrato nel [Capitolo 2.1](#_2.1_L’algoritmo), le features più rilevanti nella classificazione, calcolate tramite la funzione ***feature\_importances\_,*** risultano essere: ***age, education, occupation, native country*** e infine ***race.*** Abbiamo proceduto quindi ad eliminare le altre features, o meglio a considerare solo quelle qui listate nella classificazione del dataset.

Per eliminare le colonne e preparare i dataset abbiamo quindi utilizzato il seguente codice:



Dopo aver eliminato queste colonne, le nuove percentuali di importanza per le features rimaste risultano essere:



La distanza media risulta essere invece:



Per quanto invece riguarda i test, otteniamo la seguente accuratezza:

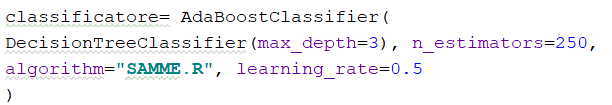


La ***Roc Curve*** e la ***Confusion matrix*** risultano essere quantomeno simili a quelle precedenti, salvo in alcuni casi. Come si evince la features selection non ha apportato miglioramenti all’accuratezza del nostro algoritmo, ma non ha causato una diminuzione eccessiva della stessa, ciò è probabilmente dovuto alla scarsità di features utilizzate per effettuare la classificazione, che risultano comunque essere la metà di quelle originariamente utilizzate, ovvero 5.

### 2.3.2 AdaBoost

AdaBoost è un algoritmo che si basa ricorsivamente su le istanze di training che il predittore precedente non ha sapute predire efficacemente (underfitting), per focalizzarsi sulle stesse e produrre un predittore ogni volta più efficiente. Quest’ algoritmo può essere applicato in combinazione con altri algoritmi provenienti dalla data science, in questo caso è stato utilizzato per ottimizzare i risultati ottenuti con random forest.

Il codice che abbiamo utilizzato per la sua applicazione è il seguente:



Anche in questo caso i risultati che abbiamo ottenuto non sono stati migliori dei precedenti, pur non peggiorando, infatti l’accuratezza è risultata essere sempre dello ***0.77***:



La mancanza di miglioramento anche qui è probabilmente da denotare alla scarsa quantità di dati utilizzati per il training e dall’impossibilità quindi di desumere un algoritmo di classificazione più prestante dagli stessi.

# 3. Sviluppi futuri

Per quanto riguarda il nostro algoritmo, abbiamo pensato a diversi modi tramite i quali questo potrebbe essere migliorato in futuro. Prima di tutto sarebbe necessario aumentare il campione di training (ovvero il dataset da noi utilizzato), cosi da allenare un predittore più efficiente rispetto a quello da noi ultimato; potremmo poi utilizzare altre metodologie di **boosting,** come Il ***GradientBoosting,*** cosi da testare l’eventuale miglioramento ottenuto grazie al suo lavorare sugli errori residui prodotti dai predittori precedenti, a dispetto di AdaBoost che fa lo stesso ma lavorando invece sui pesi da essi assegnati; infine potremmo effettuare un **tweaking** dei parametri presi in input dai vari algoritmi, come il numero di estimatori, di jobs, la profondità degli alberi ecc.

Questi sono i principali aspetti di cui riteniamo sia fondamentale lo studio, al fine di perfezionare ulteriormente lo script da noi prodotto.

Per quanto riguarda invece la re-identificazione in generale, il nostro parere a riguardo è che poiché la maggior parte degli studi effettuati in questo ambito sono nel contesto dei social network, sarebbe importante spostarsi anche su dataset che potranno poi essere d’aiuto in ambito di sicurezza informatica, in particolare per protezione di dati sensibili, provenienti ad esempio da e-shop, sistemi di pagamento online, mail-list ecc; poiché anche in quei contesti esistono dati pubblici anonimizzati a cui può essere associata un identità con il giusto algoritmo; e trovare una tecnica efficiente a questo proposito, è il primo passo per poi implementare metodi di sicurezza più efficaci.

Una delle mancanze a questo riguardo è la scarsità di utilizzo di reti neurali nelle problematiche di questo tipo, durante la nostra ricerca infatti, ci siamo imbattuti per lo più in studi che utilizzavano random forest, o grafi per portare a termine l’identificazione degli individui.

# 4. Conclusioni

Alla luce dello sviluppo, del testing e delle altre strade che abbiamo percorso per migliorare l’accuratezza del nostro progetto, possiamo ritenerci soddisfatti di ciò che abbiamo realizzato. Nonostante l’accuratezza risultata non sia elevatissima, dobbiamo pur tenere conto della grandezza del dataset da noi preso in considerazione e soprattutto dall’aggiunta del rumore nei dati che ha limitato fortemente le possibilità di desumere un predittore migliore durante le varie iterazioni degli algoritmi che abbiamo utilizzato, che si parli di random forest, features selection o AdaBoost.

In conclusione, grazie a questo progetto abbiamo potuto appassionarci all’argomento della re-identificazione e siamo molto appagati dal risultato, pur non essendo rivoluzionario o molto importante, siamo felici di aver fatto fare un passettino avanti a questa branchia della data science, che speriamo venga presa sempre più in considerazione in futuro.

# Riferimenti

|  |  |
| --- | --- |
| [1] | Wikipedia, «Random Forest,» [Online]. Available: https://en.wikipedia.org/wiki/Random\_forest. |
| [2] | Wikipedia, «Data\_re-identification,» [Online]. Available: https://en.wikipedia.org/wiki/Data\_re-identification. |
| [3] | T. Dalenius, "Finding a Needle in a Haystack," *Journal of official statistics,* vol. 2, no. 3, p. 9, 1986. |
| [4] | J. Ma, «De-Anonymizing Social Networks With Random Forest Classifier,» IEEE Xplore, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8051053. |
| [5] | J. Fang, "A Structure-Based De-Anonymization Attack on Graph Data Using Weighted Neighbor Match," IEEE, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8923567. |
| [6] | U. -m. learning, «UCI - machine learning repository,» [Online]. Available: https://archive.ics.uci.edu/ml/index.php. |
| [7] | Wikipedia, «Record Linkage,» [Online]. Available: https://en.wikipedia.org/wiki/Record\_linkage. |
| [8] | DataRobot, «Model Fitting,» [Online]. Available: https://www.datarobot.com/wiki/fitting/. |
| [9] | Wikipedia, «Distanza Euclidea,» [Online]. Available: https://it.wikipedia.org/wiki/Distanza\_euclidea. |
| [10] | S. Narkhede, «Understanding AUC - ROC Curve,» 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. |
| [11] | GeeksforGeeks, «Confusion Matrix in Machine Learning,» [Online]. Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/. |