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

Abstract

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.
In particolare discuteremo dell'utilizzo di algoritmi di Machine Learning, come Random Forest in questo ambito e dei risultati prodotti in questo studio.
Dopo aver importato e ripulito un dataset ed averne creato uno ad-hoc per effettuare poi il matching abbiamo prodotto uno script capace di associare ai soggetti anonimizzati le identità ivi contenute, infine, siamo passati al testing delle performance dello script attraverso metodi appositi (Roc Curve & Confusion Matrix) e al miglioramento delle stesse attraverso metodi di Boosting (feature selection & AdaBoost).

Introduzione

La tematica della re-identificazione risulta essere di interesse in campi che spaziano da quello finanziario a quello della salute poiché moltissime aziende accumulano e rendono poi pubblici dati di soggetti 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; e riuscire re-identificare questi dati è il primo passo verso il miglioramento delle tecniche di de-identificazione e di conseguenza la protezione degli stessi.
Le principali tecniche atte alla rimozione/mascheramento dei dati, sono composte per lo più da processi di masking, generalizzazione o cancellazione. 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 particolare con metodologie di machine learning, per effettuare 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.

Di seguito tratteremo di come, utilizzando un dataset prelevato dal sito web **UCI machine learning**, abbiamo, attraverso l'utilizzo di algoritmi di ***Random Forest*** [1] e di identità create ad-hoc, effettuato una re-identificazione sui dati, sfruttando le diverse feature (dati per ogni individuo) presenti nel dataset ivi prelevato.

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.

1.1 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 state prese in considerazione 4 alternative: distruggere i record contenenti i dati unici rilevati; rendere vuoti i campi riguardanti i dati unici rilevati; utilizzare qualche tecnica di perturbazione dei dati o infine cifrare 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 feature 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.

1.4 "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 feature 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, utilizza le funzioni ottenute dal metodo "weighted neighborhood" e una matrice di similarità dinamica, per procedere poi a calcolare la similarità tra grafi anonimi ed ausiliari, impiegando come detto in precedenza features locali e globali.
Tra le conclusioni si è 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 **feature**) per ogni soggetto risultano essere:

Listing of attributes:

>50K, <=50K.

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Il dataset composto quindi da ***14 attributi e 32561 soggetti***, è stato da noi sottoposto 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 feature 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:

	A	B	C	D	E	F	G	H	I	J	K
1	Age	Workclass	Education	Education-num	Marital-status	Occupation	Relationship	Race	Sex	Native-Country	Income
2	27	Private	Some-college	10	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	Cambodia	>50K
3	48	Private	Some-college	10	Married-civ-spouse	Craft-repair	Other-relative	Asian-Pac-Islander	Male	Cambodia	>50K
4	45	Private	Some-college	10	Married-civ-spouse	Adm-clerical	Wife	White	Female	Canada	>50K
5	37	Local-gov	Some-college	10	Married-civ-spouse	Adm-clerical	Husband	White	Male	Canada	>50K
6	36	Private	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	Canada	>50K
7	32	Private	Some-college	10	Married-civ-spouse	Handlers-cleaners	Husband	White	Male	Canada	>50K
8	55	Private	Some-college	10	Married-civ-spouse	Tech-support	Husband	White	Male	Canada	>50K
9	52	Private	Some-college	10	Married-civ-spouse	Sales	Wife	White	Female	Canada	>50K
10	56	Self-emp-inc	Some-college	10	Married-civ-spouse	Sales	Wife	White	Female	Canada	>50K
11	59	Private	Some-college	10	Married-civ-spouse	Adm-clerical	Wife	White	Female	Cuba	>50K
12	33	Private	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	White	Male	Cuba	>50K
13	31	Private	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	White	Male	Cuba	>50K
14	38	Self-emp-inc	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	Cuba	>50K
15	37	Private	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	Cuba	>50K
16	27	Self-emp-not-inc	Some-college	10	Married-civ-spouse	Transport-moving	Other-relative	Other	Male	Ecuador	>50K
17	42	Private	Some-college	10	Married-civ-spouse	Exec-managerial	Wife	White	Female	El-Salvador	>50K
18	43	Private	Some-college	10	Married-civ-spouse	Other-service	Wife	White	Female	England	>50K
19	34	Private	Some-college	10	Never-married	Machine-op-inspct	Not-in-family	White	Male	England	>50K

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, indirizzo e data di nascita, opportunamente inventati ad-hoc per rispecchiare i soggetti anonimizzati ai quali si riferivano:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Name	Birthday	Address (CAP)	Age	Workclass	Education	Marital-status	Occupation	Relationship	Race	Sex	Native-Country	Income
2	Adolf	09/12/1995	22489 , Germany	37	Private	Some-college	Married-civ-spouse	Adm-clerical	Husband	Asian-Pac-Islander	Male	Canada	>50K
3	Margareth	05/10/1960	4299, Phillipines	31	Local-gov	Masters	Never-married	Other-service	Other-relative	White	Female	England	>50K
4	Benny	28/04/1965	7039, United-States	45	Self-emp-inc	Prof-school	Divorced	Tech-support	Not-in-family	White	Male	Germany	>50K
5	Truman	18/09/1990	25813, United-State	63	Federal-gov	HS-grad	Widowed	Farming-fishing	Not-in-family	Black	Male	Iran	>50K
6	Rudolf	31/01/1997	30381, United-State	21	Self-emp-not-inc	Bachelors	Widowed	Craft-repair	Not-in-family	Other	Male	Jamaica	>50K
7	Jenny	11/03/1950	94209, United-States	35	Private	Assoc-voc	Divorced	Machine-op-inspct	Not-in-family	Asian-Pac-Islander	Female	United-States	<=50K
8	Laura	03/01/1981	84015, Italy	66	Local-gov	Assoc-acdm	Never-married	Adm-clerical	Not-in-family	Black	Female	Mexico	<=50K
9	Silvia	20/04/1971	58282, United-States	63	Self-emp-inc	7th-8th	Married-civ-spouse	Exec-managerial	Wife	Other	Female	Peru	<=50K
10	Hans	06/07/1999	22589 , Germany	39	Federal-gov	9th	Married-civ-spouse	Sales	Husband	White	Male	Philippines	<=50K
11	Olivia	03/05/1975	04100, Italy	62	Self-emp-not-inc	10th	Married-civ-spouse	Prof-specialty	Wife	White	Female	United-States	>50K
12	Jean	25/09/1978	22489, Austria	41	Private	Bachelors	Married-civ-spouse	Handlers-cleaners	Husband	White	Male	Germany	<=50K
13	Anne	09/12/1995	4299, France	24	Private	Bachelors	Never-married	Priv-house-serv	Not-in-family	White	Female	France	<=50K
14	Veronica	12/06/1985	7039, Italy	34	State-gov	Bachelors	Married-civ-spouse	Adm-clerical	Husband	White	Male	Italy	<=50K
15	Mario	18/09/1985	25813, Italy	34	State-gov	Bachelors	Married-civ-spouse	Adm-clerical	Husband	White	Male	Italy	<=50K
16	Gaspare	31/07/1965	30381, United-States	54	Self-emp-not-inc	Bachelors	Married-civ-spouse	Sales	Husband	White	Male	United-States	>50K
17	Susenne	23/01/1971	94209, United-States	48	Private	Bachelors	Divorced	Other-service	Unmarried	White	Female	United-States	>50K
18	Ashen	13/03/1992	84015, France	27	Private	Bachelors	Never-married	Tech-support	Not-in-family	Black	Female	France	<=50K
19	Sonia	22/10/1976	58282, United-States	43	Private	10th	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	United-States	>50K
20	Abdul	26/09/1994	22589, Africa	25	Private	10th	Never-married	Other-service	Not-in-family	White	Male	Nicaragua	<=50K
21	Cameron	14/10/1962	4100, United-States	55	Private	10th	Married-civ-spouse	Craft-repair	Husband	White	Male	United-States	>50K
22	Gregory	07/11/1976	7101, United-States	43	Local-gov	Masters	Never-married	Prof-specialty	Unmarried	White	Male	United-States	>50K
23	Marianna	21/09/1998	118, Italy	21	Private	Bachelors	Never-married	Farming-fishing	Unmarried	White	Female	Italy	<=50K
24	Foreman	03/01/1985	5664, United-States	34	Local-gov	Masters	Married-civ-spouse	Prof-specialty	Husband	Black	Male	France	>50K
25	Taub	08/05/1978	881, United-States	41	Local-gov	Masters	Divorced	Prof-specialty	Not-in-family	White	Male	England	>50K
26	Marco	08/12/1995	234, Italy	24	Private	HS-grad	Never-married	Sales	Not-in-family	Black	Male	Peru	<=50K
27	Lucas	04/11/1994	730, France	25	Self-emp-not-inc	Some-college	Divorced	Handlers-cleaners	Not-in-family	White	Male	Germany	<=50K

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 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, quella che quindi, più si avvicinasse ad una tra quelle presenti 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:

```
COLUMNS = ['Age', 'Workclass', 'Education',    'Marital-status', 'Occupation', 'Relationship',  'Race',
           'Sex', 'Native-Country', 'Income']

features_per_allenamento = pd.read_csv (r'C:\\Users\\FauxL\\Desktop\\Data Science\\Project\\training.csv',
                                        usecols = COLUMNS)
features_per_predire = pd.read_csv (r'C:\\Users\\FauxL\\Desktop\\Data Science\\Project\\adult.csv',
                                    usecols = COLUMNS)
features_per_encoder = features_per_predire
```

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 stringhe, ed i nostri dataset erano formati in gran parte da esse, abbiamo proceduto, quindi, 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, l'encoder è stato prima allenato sul dataset adult, contenente quindi tutte le possibili stringhe per ogni feature, e poi applicato su entrambi i dataset trasformando le strighe in semplici valori numerici:

```
# fitto il LabelEncoder con tutte le features usando adult.csv che contiene tutte le possibili classi per ogni feature
label_encoder = preprocessing.LabelEncoder()
dataframe_predizione = pd.DataFrame()
dataframe_training = pd.DataFrame()
```

```
# For da 0 a 9 per trasformare gli array di features da Stringhe ad numeri Float
for x in range(0,9):
    singola_feature_predizione = features_per_predire.iloc[:, x]
    singola_feature_training = features_per_allenamento.iloc[:, x]
    singola_feature_encoder = features_per_encoder.iloc[:, x]

    label_encoder.fit(singola_feature_encoder)  # fitto l'encoder con tutte le possibili classi della feature

    # presa una colonna di feature, normalizzo i dati della feature del dataset adult.csv
    singola_feature_predizione = pd.DataFrame(label_encoder.transform(singola_feature_predizione))
    dataframe_predizione = pd.concat([dataframe_predizione, singola_feature_predizione], axis=1)

    label_encoder.fit(singola_feature_encoder)  # fitto l'encoder con tutte le possibili classi della feature, DI NUOVO

    # per la stessa feature, normalizzo i dati del dataset di training.csv
    singola_feature_training = pd.DataFrame(label_encoder.transform(singola_feature_training))
    dataframe_training = pd.concat([dataframe_training, singola_feature_training], axis=1)
```

Dopo aver quindi ottenuto dei dataset con valori numerici procediamo a creare un **classificatore**, con **250 estimatori**, con criterio **entropy**, importando la colonna ***Name*** usate come **label** per poter effettuare il **fitting** [8]***:***

```
classi_target = pd.read_csv(r'C:\\Users\\FauxL\\Desktop\\Data Science\\Project\\training.csv', usecols=['Name'])
classificatore = RandomForestClassifier(n_estimators = 250, criterion='entropy', n_jobs=4)
classificatore.fit(dataframe_training, classi_target.values.ravel()) # fitto con tutto il dataset di training
```

Otteniamo i primi dati utili da quanto realizzato, in particolare si tratta della rilevanza di ogni features nel processo di identificazione, di seguito mostrata:

```
print("\nRilevanza attributi")
for name, score in zip(COLUMNS, classificatore.feature_importances_):
    print(name, score)
```

```
Rilevanza attributi
Age 0.22276457564738789
Workclass 0.058377560516826035
Education 0.11675034678995218
Marital-status 0.08008379492063797
Occupation 0.15294655669870136
Relationship 0.06939819997774646
Race 0.09743037106018078
Sex 0.043115914238977746
Native-Country 0.1591326801495895
```

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:

```
# testo il classificatore con delle classi che già conosce e di cui conosce anche l array di feature
# per testarne la precisione
predict = classificatore.predict(dataframe_training)
predict = np.array(predict)
classi_target = list(np.array(classi_target))

# Predico utilizzando il dataset dell'esperimento
predizione = classificatore.predict(dataframe_predizione)
```

Dopo aver effettuato la predizione abbiamo deciso 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:

```
#calcolo per ogni risultato la distanza euclidea dall'array predetto
dataframe_risultati = pd.DataFrame(predizione)

# prelevo gli array obiettivo dal dataset training.csv e li assegno
array_name=  pd.read_csv (r'C:\\Users\\FauxL\\Desktop\\Data Science\\Project\\training.csv', usecols = ["Name"]).to_numpy()

risultati_con_distanza = pd.DataFrame()
arrayDistanze = [ ]

# For per il calcolo delle distanze euclidee
somma_tot_distanze = 0
for y in range(1, dataframe_risultati.size):
    nome_predetto = dataframe_risultati.iloc[y].values # prendo l identità predetta dell' y-esimo elemento
    array_features = dataframe_predizione.iloc[y] # prendo l' array di features dell' y-esimo elemento
    j=0
    while j < len(array_name):
        if nome_predetto == array_name[j]:
            array_obiettivo = dataframe_training.iloc[j]
            break
        j=j+1

    dst = distance.euclidean(array_features, array_obiettivo)
    nome_predetto = pd.DataFrame(dataframe_risultati.iloc[y])

    somma_tot_distanze = somma_tot_distanze + dst
    # print('elemento n [' , y, '] con identità ' , nome_predetto.values, ' con distanza: [' , dst , ']
    # da array obiettivo: ', array_obiettivo.values  )
    arrayDistanze.append(dst)
```

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 assegnata 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:

```
# conto il numero di predizione per ogni classe
np.set_printoptions(threshold = sys.maxsize)
unici, counteggio = np.unique(dataframe_risultati, return_counts=True)
print(unici,counteggio)

# do in output la distanza media come indice di errore per testare
print('valore medio delle distanze: ', somma_tot_distanze/dataframe_risultati.size)
```

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 all'algoritmo di adattarsi troppo al dataset di addestramento permettendo allo stesso di lavorare meglio sui dati; alla luce di ciò, l'output del codice di cui sopra risulta essere:

```
['Abdul' 'Adolf' 'Alexa' 'Alien' 'Andrek' 'Anne' 'Annina' 'Ashen' 'Avio'
 'Barack' 'Benny' 'Bianca' 'Caddy' 'Cameron' 'Carmela' 'Deadman' 'Deuz'
 'Ebry' 'Emma' 'Ethas' 'Foreman' 'Fragile' 'Gaspare' 'Ghandi' 'Ginius'
 'Gino' 'Gioggiorgio' 'Giovanna' 'Giulhio' 'Giuseppina' 'Goldman'
 'Gregory' 'Hans' 'Hardman' 'Helio' 'Hideo' 'Higs' 'Hiremo' 'Hulio'
 'Ilenia' 'Istine' 'Jean' 'Jenny' 'Kairi' 'Kirale' 'Kojima' 'Laura'
 'Lella' 'Leona' 'Lillia' 'Lisa' 'Lucas' 'Malone' 'Mama' 'Marcelo' 'Marco'
 'Margareth' 'Marianna' 'Mario' 'Mister' 'Nekonami' 'Obama' 'Oleos'
 'Olivia' 'Orend' 'Otella' 'Palee' 'Paul' 'Pepp' 'Pina' 'Posterina'
 'Prapapaola' 'Raiden' 'Rita' 'Rudolf' 'Sam' 'Settimio' 'Silvia' 'Siria'
 'Sirius' 'Sonia' 'SufferMan' 'Susenne' 'Taub' 'TrueMan' 'Truman'
 'Valerio' 'Veronica' 'Violo' 'Virginio' 'Wile' 'Wongo'] [1215   36  427  229   34  213   12   30 1879   43   30   10    6 1426
  243  151  115   18   36   22   30   15 1413  626  282   30   97   73
  561   12   43  356 1055 1453    6    8   31    9   14    3 1517  540
  696  131   65   70  225   24    2  242  801  121 3469  110  164  198
   44 2308   28  126  662   12  894  107  504    4   45  127  178  309
   18   41   98  115    2  559   35  143    6   14  102  186  687    3
   17    2    1  404  609  618  378  109]
valore medio delle distanze:  12.407201193543704
```

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

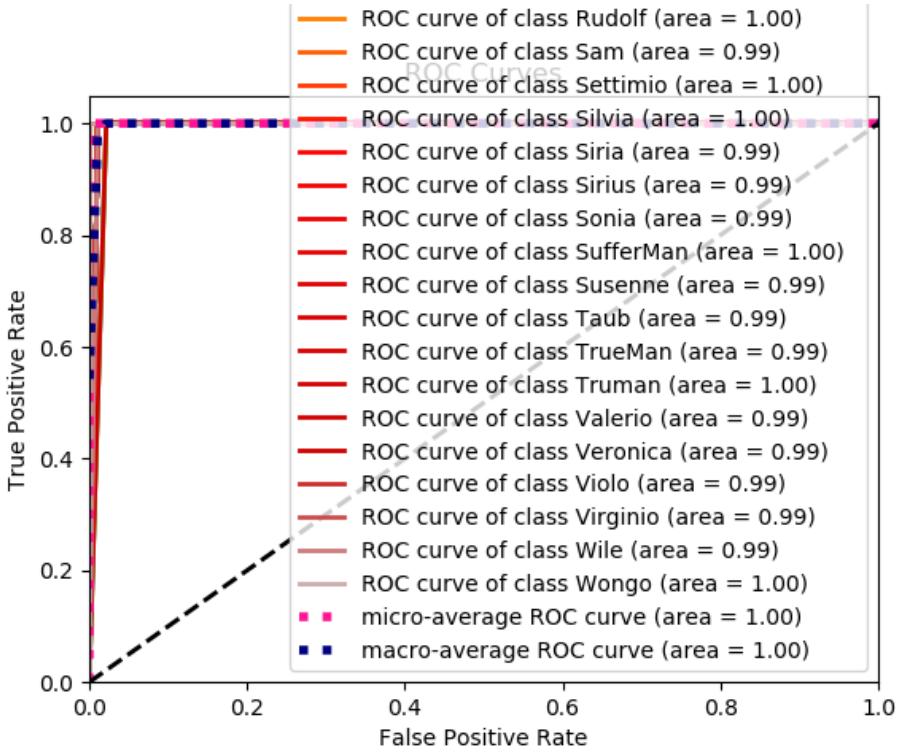
Come primo passo abbiamo definito una funzione per la creazione e il plotting della Roc Curve:

```
def plot_roc_curve(fpr, tpr):
    x = fpr
    y = tpr
    plt.plot(x, y, color='orange', linestyle='solid' )
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='solid')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

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

```
# Utilizzo la funzione plot_roc_curve() definita sopra per disegnare il grafico della roc curve
skplt.metrics.plot_roc(classi_target, predict_proba)
plt.show()
```

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:

```
cnf_matrix = confusion_matrix(classi_target, predict)
print(' - Confusion Matrix -')
print(cnf_matrix)
print(' - Accuracy Score -', accuracy_score(classi_target, predict))
print(' - Report  -'), print(classification_report(classi_target, predict))

FP = cnf_matrix.sum(axis=0) - np.diag(cnf_matrix)
FN = cnf_matrix.sum(axis=1) - np.diag(cnf_matrix)
TP = np.diag(cnf_matrix)
TN = cnf_matrix.sum() - (FP + FN + TP)

FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

FP = cnf_matrix.sum(axis=0) - np.diag(cnf_matrix)
FN = cnf_matrix.sum(axis=1) - np.diag(cnf_matrix)
TP = np.diag(cnf_matrix)
TN = cnf_matrix.sum() - (FP + FN + TP)

FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
# False discovery rate
FDR = FP/(TP+FP)
# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
```

Otteniamo il seguente output:

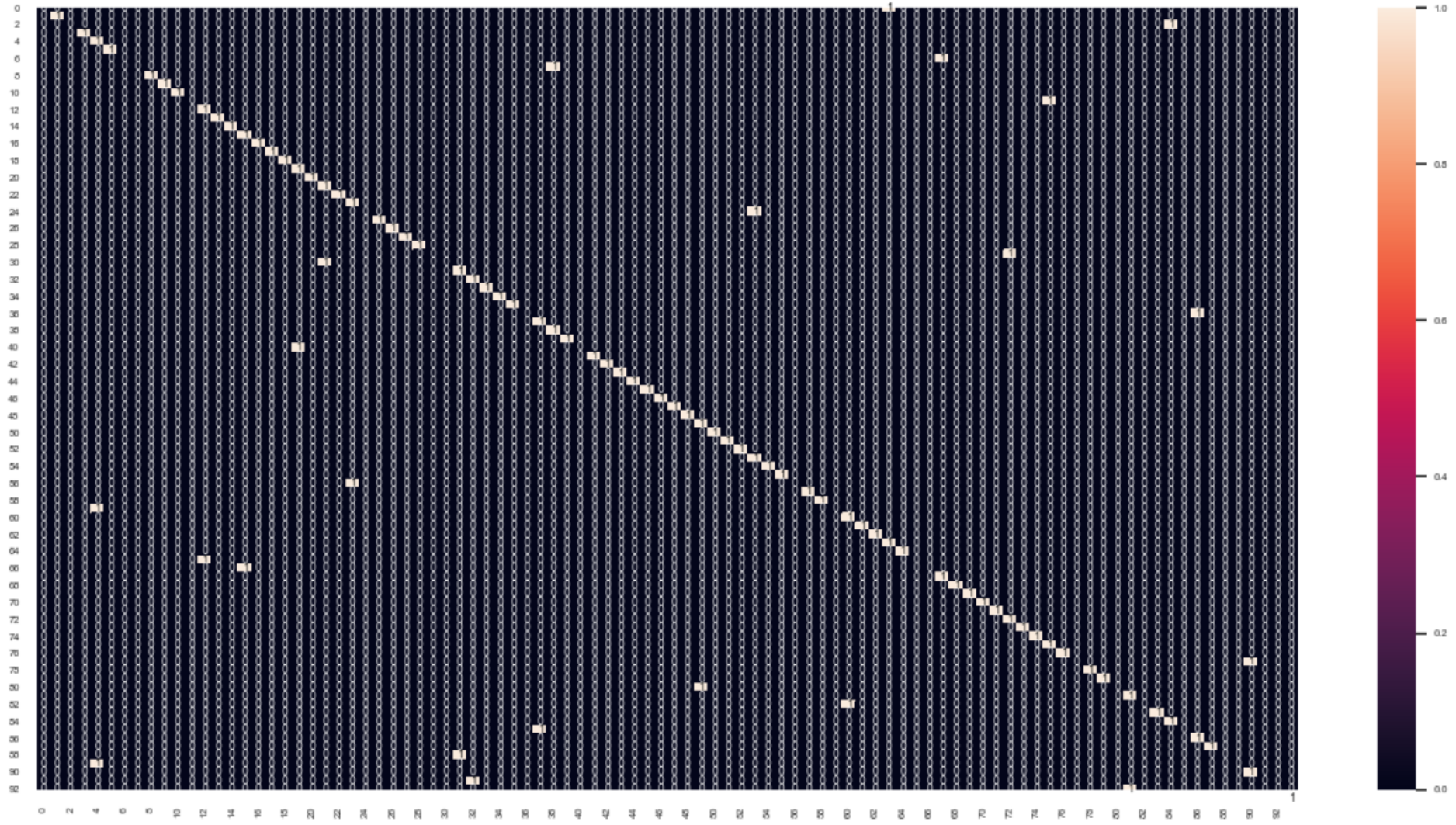
- Confusion Matrix -

```
[[0 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 1]]
```

- Accuracy Score - 0.7659574468085106

- Report -

	precision	recall	f1-score	support
Abdul	0.00	0.00	0.00	1
Adolf	1.00	1.00	1.00	1
Alexa	0.00	0.00	0.00	1
Alien	1.00	1.00	1.00	1
Andrek	0.33	1.00	0.50	1
Anne	1.00	1.00	1.00	1
Annina	0.00	0.00	0.00	1
Ashen	0.00	0.00	0.00	1
Avio	1.00	1.00	1.00	1
Barack	1.00	1.00	1.00	1
Benny	1.00	1.00	1.00	1
Bianca	0.00	0.00	0.00	1
Caddy	0.50	1.00	0.67	1
Cameron	1.00	1.00	1.00	1
Carmela	1.00	1.00	1.00	1
Deadman	0.50	1.00	0.67	1
Deuz	1.00	1.00	1.00	1
Ebry	1.00	1.00	1.00	1
Emma	1.00	1.00	1.00	1
Ethas	0.50	1.00	0.67	1
Foreman	1.00	1.00	1.00	1
Fragile	0.50	1.00	0.67	1
Gaspare	1.00	1.00	1.00	1
Ghandi	0.50	1.00	0.67	1
Ginius	0.00	0.00	0.00	1
Gino	1.00	1.00	1.00	1
Gioggiorgio	1.00	1.00	1.00	1
Giovanna	1.00	1.00	1.00	1
Giulhio	1.00	1.00	1.00	1
Giuseppina	0.00	0.00	0.00	1
Goldman	0.00	0.00	0.00	1
Gregory	0.50	1.00	0.67	1
Hans	0.50	1.00	0.67	1
Hardman	1.00	1.00	1.00	1
Helio	1.00	1.00	1.00	1
Hideo	1.00	1.00	1.00	1
Higs	0.00	0.00	0.00	1
Istine	1.00	1.00	1.00	1
Jean	1.00	1.00	1.00	1
Jenny	1.00	1.00	1.00	1
Kairi	1.00	1.00	1.00	1
Kirale	1.00	1.00	1.00	1
Kojima	1.00	1.00	1.00	1
Laura	1.00	1.00	1.00	1
Lella	1.00	1.00	1.00	1
Leona	0.50	1.00	0.67	1
Lillia	1.00	1.00	1.00	1
Lisa	1.00	1.00	1.00	1
Lucas	1.00	1.00	1.00	1
Malone	0.50	1.00	0.67	1
Mama	1.00	1.00	1.00	1
Marcelo	1.00	1.00	1.00	1
Marco	0.00	0.00	0.00	1
Margareth	1.00	1.00	1.00	1
Marianna	1.00	1.00	1.00	1
Mario	0.00	0.00	0.00	1
Mister	0.50	1.00	0.67	1
Nekonami	1.00	1.00	1.00	1
Obama	1.00	1.00	1.00	1
Oleos	0.50	1.00	0.67	1
Olivia	1.00	1.00	1.00	1
Oreck	0.00	0.00	0.00	1
Orend	0.00	0.00	0.00	1
Otella	0.50	1.00	0.67	1
Palee	1.00	1.00	1.00	1
Paul	1.00	1.00	1.00	1
Pepp	1.00	1.00	1.00	1
Pina	1.00	1.00	1.00	1
Posterina	0.50	1.00	0.67	1
Prapapaola	1.00	1.00	1.00	1
Raiden	1.00	1.00	1.00	1
Rita	0.50	1.00	0.67	1
Rudolf	1.00	1.00	1.00	1
Sam	0.00	0.00	0.00	1
Settimio	1.00	1.00	1.00	1
Silvia	1.00	1.00	1.00	1
Siria	0.00	0.00	0.00	1
Sirius	0.50	1.00	0.67	1
Sonia	0.00	0.00	0.00	1
SufferMan	1.00	1.00	1.00	1
Susenne	0.50	1.00	0.67	1
Taub	0.00	0.00	0.00	1
TrueMan	0.50	1.00	0.67	1
Truman	1.00	1.00	1.00	1
Valerio	0.00	0.00	0.00	1
Veronica	0.00	0.00	0.00	1
Violo	0.50	1.00	0.67	1
Virginio	0.00	0.00	0.00	1
Wile	0.00	0.00	0.00	1
Wongo	1.00	1.00	1.00	1
accuracy			0.77	94
macro avg	0.65	0.77	0.69	94
weighted avg	0.65	0.77	0.69	94


0
2
4
6
5
10
12
14
16
15
20
22
24
26
25
30
32
34
36
35
40
42
44
46
45
50
52
54
56
55
60
62
64
66
65
70
72
74
76
75
80
82
84
86
85
90
92
0
2
4
6
5
10
12
14
16
15
20
22
24
26
25
30
32
34
36
35
40
42
44
46
45
50
52
54
56
55
60
62
64
66
65
70
72
74
76
75
80
82
84
86
85
90
92
1
1
1.0
0.5
0.6
0.4
0.2
0.0

Quella sopra riportata è la confusion matrix, se tutte le label fossero correttamente assegnate la diagonale dovrebbe essere composta soltanto da 1, nel nostro caso il classificatore etichetta correttamente un soggetto nel 77% dei casi, ciò si evince dalla percentuale di accuratezza restituita dalla C.M. che è ***0.77***.

Nel nostro percorso di testing con dataset piccoli e composti da soggetti molto diversi tra loro l'algoritmo era capace di distinguere ed etichettare con accuratezza molto vicina ad 1, ovvero il 100% dei casi; in casi reali e di impiego nel mondo reale, tuttavia, ci è sembrato irrealistico che il dataset di training fosse di così ridotte dimensioni e soprattutto con individui molto diversi tra loro. Abbiamo quindi aggiunto molti più individui al dataset e, sopratutto, aggiunto molto più rumore, ovvero individui identici tra loro *(feature uguali*) oppure individui molto simili tra loro (*quasi tutte le feature identiche tranne una*) ma con identità diverse (*nome diverso*).

Ciò ha portato, quindi, ad una minore accuratezza dell'algoritmo ma ci ha permesso di capire come l'algoritmo opera in una situazione molto più realistica rispetto alle fasi di testing iniziali.

Il codice che abbiamo utilizzato per il plot è il seguente:

```
print(type(classi_target))
df_cm = pd.DataFrame(cnf_matrix)
sn.set(font_scale=0.5) # for label size
sn.heatmap(df_cm, annot=True) # font size
plt.show()
```

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 feature 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, le feature 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 feature, 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:

```
pin = [ 'Workclass', 'Marital-status', 'Relationship',  'Sex','Income' ]

features_per_allenamento = features_per_allenamento.drop(columns=pin[0:5])
features_per_predire = features_per_predire.drop(columns=pin[0:5])
features_per_encoder = features_per_encoder.drop(columns=pin[0:5])
```

Dopo aver eliminato queste colonne, abbiamo ricalcolato ***feature_importances_*** e i risultati risultano essere:

```
Rilevanza attributi
Age 0.2922489224773368
Education 0.1614697757200497
Occupation 0.20415141583190202
Race 0.12842996077067595
Native-Country 0.21369992520003556
```

La distanza media risulta essere invece:

```
valore medio delle distanze:  10.675991034253203
```

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

```
- Accuracy Score - 0.7021276595744681
```

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 feature 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 saputo predire efficacemente (underfitting), per focalizzarsi sulle stesse e produrre un predittore ogni volta più efficiente.

Tale algoritmo è stato utilizzato per ottimizzare i risultati ottenuti con l'algoritmo ***Random Forest***.

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

```
classificatore= AdaBoostClassifier(
DecisionTreeClassifier(max_depth=3), n_estimators=250,
algorithm="SAMME.R", learning_rate=0.5
)
```

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 ***0.765***:

```
- Accuracy Score - 0.7659574468085106
```

La mancanza di miglioramento anche qui è probabilmente da denotare alla scarsa quantità di dati utilizzati per il training (*altre features per ogni individuo*) e dall'impossibilità, quindi, di desumere un algoritmo di classificazione più prestante dagli stessi; in sostanza si riconferma la teoria della *Data Science* secondo il quale "nella branca del *Machine Learning* più **dati** aiutano più di un **algoritmo** migliore"

3. Sviluppi futuri

Per quanto riguarda il nostro progetto, ci sono diversi modi tramite i quali potrebbe essere migliorato in futuro.

Innanzitutto 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.

Potrebbero essere utilizzate altre metodologie di **boosting,** come Il ***GradientBoosting,*** cosi da testare l'eventuale miglioramento ottenuto grazie al suo lavoro sugli errori residui prodotti dai predittori precedenti, a dispetto di AdaBoost che fa lo stesso ma lavorando invece sui pesi da essi assegnati.

Infine si potrebbe effettuare un **tweaking** dei parametri presi in input dai vari algoritmi, come il numero di estimatori, di jobs, la profondità degli alberi ecc. e rieffettuare tutti i test e confrontarli con quelli da noi prodotti.

Questi sono i principali aspetti di cui riteniamo sia fondamentale lo studio, al fine di perfezionare ulteriormente l'algoritmo da noi prodotto ed approfondire lo studio da noi affrontato.

Per quanto riguarda invece l'argomento di *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 di 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, siamo felici di aver fatto fare un altro passo avanti a questa branca della *Data Science* e 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/.