

# Report Processo CRISP-DM

Gennaro Sisto - Mat: 705834

<b>Introduzione</b>	<b>4</b>
<b>1. Business Understanding</b>	<b>4</b>
<b>2. Data mining tasks</b>	<b>4</b>
2.1 Predictive data mining task	4
2.2 Descriptive data mining task	4
<b>3. Data Understanding</b>	<b>5</b>
3.1 dataset PoliceKillingsUS	5
3.2 dataset stati americani	15
3.3 dataset PercentOver25CompletedHighSchool	16
3.4 dataset PercentagePeopleBelowPovertyLevel	17
3.5 dataset MedianHouseholdIncome2015	18
3.6 dataset ShareRaceByCity	19
<b>4. Data Preparation</b>	<b>20</b>
<b>4.1 dataset PoliceKillingsUS</b>	<b>20</b>
4.1.1 Caricamento in memoria del dataset	20
4.1.2 Eliminazione attributi	20
4.1.3 Attributo age	21
4.1.4 Attributo city	21
4.1.5 Attributo state	22
4.1.6 Attributo armed	22
4.1.7 Attributo flee	23
4.1.8 Attributo threat_level	24
4.1.9 Attributo race	25
4.1.10 Attributo signs_of_mental_illness	26
4.1.11 Attributo body_camera	27
<b>4.2 dataset stati americani</b>	<b>28</b>
4.2.1 Uso di Power Query di Excel	28
4.2.2 Uso di Python	33
4.2.2.1 Elaborazioni sui singoli dataset	33
4.2.2.2 Merge dei dataset americani	34
<b>4.3 Prima integrazione</b>	<b>35</b>
<b>4.4 dataset PercentOver25CompletedHighSchool</b>	<b>36</b>
4.4.1 Caricamento in memoria del dataset	36

4.4.2 Sostituzione valori	36
4.4.3 Formattazione nuovo attributo	37
<b>4.5 Seconda integrazione</b>	<b>38</b>
<b>4.6 dataset PercentagePeopleBelowPovertyLevel</b>	<b>39</b>
4.6.1 Caricamento in memoria del dataset	39
4.6.2 Sostituzione valori	39
4.6.3 Formattazione nuovo attributo	40
<b>4.7 Terza integrazione</b>	<b>41</b>
<b>4.8 dataset MedianHouseholdIncome2015</b>	<b>42</b>
4.8.1 Caricamento in memoria del dataset	42
4.8.2 Sostituzione valori	42
4.8.3 Renaming attributo Median Income	43
4.8.4 Formattazione nuovo attributo	43
<b>4.9 Quarta integrazione</b>	<b>44</b>
<b>4.10 dataset ShareRaceByCity</b>	<b>45</b>
4.10.1 Caricamento in memoria del dataset	45
4.10.2 Sostituzione valori	45
4.10.3 Creazione nuovo attributo <i>share_not_black</i>	46
4.10.4 Formattazione nuovo attributo	46
<b>4.11 Quinta e ultima integrazione</b>	<b>47</b>
<b>5. Modeling</b>	<b>48</b>
<b>5.1 Prime considerazioni</b>	<b>48</b>
5.1.1 Caricamento del dataset	48
5.1.2 Selezione attributo di classe	49
5.1.3 Applicazione filtro SpreadSubsample	50
<b>5.2 Prediction</b>	<b>51</b>
5.2.1 Elaborazioni finali	51
5.2.1.1 Discretization	51
5.2.1.1 Divisione del dataset e avvio procedura	54
5.2.2 K-NN	55
5.2.3 C4.5 Classification Tree – J48	57
5.2.4 Naive Bayes	59
5.2.5 Ripper – JRipper	61
<b>5.3 Description</b>	<b>63</b>
5.3.1 Elaborazioni finali con rimozione di attributi	63
5.3.2 K-means – SimpleKMeans	64
<b>6. Evaluation</b>	<b>67</b>
<b>6.1 Prediction evaluation</b>	<b>67</b>
<b>6.2 Description evaluation</b>	<b>67</b>
<b>7. Deployment</b>	<b>68</b>

<b>7.1 Predictive task</b>	<b>68</b>
<b>7.2 Descriptive task</b>	<b>68</b>

## Introduzione

Il dataset su cui si è deciso di avviare un processo di *Knowledge Discovery* è chiamato “Fatal Police Shootings in the US”, ed è reperibile a questo link <sup>[1]</sup>.

### 1. Business Understanding

Il seguente report si pone come obiettivo quello di trovare dei modelli di data mining tali da poter analizzare e descrivere un problema attuale e di grande portata: gli interventi fatali degli agenti della polizia statunitense, che hanno portato alla nascita del movimento “Black Lives Matter”.

### 2. Data mining tasks

I modelli di data mining esaminati prendono in input i seguenti parametri, considerati essenziali per poter proseguire con le analisi proposte:

- La città in cui è avvenuto l'intervento della polizia;
- Il numero di abitanti della città;
- Il reddito medio della città;
- Il tasso di povertà della città;
- La percentuale di persone che si sono diplomate, di età superiore ai 25;
- La percentuale di popolazione di etnia afroamericana;
- La percentuale di popolazione di etnia non afroamericana;
- L'età della vittima;
- Il sesso della vittima;
- Se la vittima era armata;
- Se la vittima è stata indicata come “mentalmente instabile”;
- Se la vittima ha dimostrato una minaccia per la causa dell'intervento, secondo la Polizia;
- Se la vittima ha tentato di scappare;
- Se la vittima era di etnia afroamericana.

#### 2.1 Predictive data mining task

Si è deciso di analizzare il dataset per provare a stimare se, effettivamente, negli interventi fatali degli agenti di polizia sia presente un fattore di discriminazione razziale nei confronti degli individui di etnia afroamericana, andando, in concreto, a predire se la vittima era di origine afroamericana o meno.

#### 2.2 Descriptive data mining task

Si è deciso di analizzare il dataset per provare ad individuare dei cluster, o degli schemi concreti, che emergano dai dati, per verificare la presenza di un **racial bias**. In particolare, si vuole provare a dimostrare che gli interventi fatali della polizia siano, in diversi casi, rivolti verso individui di etnia afroamericana di età non superiore ai 25-30 anni, soprattutto in grandi città dove la presenza di persone di etnia afroamericana è comunque minore rispetto al resto delle altre etnie.

---

<sup>1</sup> <https://www.kaggle.com/kwulum/fatal-police-shootings-in-the-us>

### 3. Data Understanding

I parametri citati nel [paragrafo 2](#) e utilizzati per le analisi proposte sono il risultato finale di un lavoro di pulizia, integrazione e normalizzazione di dati, effettuato su più dataset, proprio perché allo stato iniziale i dati erano presenti in forma complessivamente grezza e/o incompleta ai fini delle analisi proposte.

Per questa fase del processo di KD, come anche per le successive, si è deciso di utilizzare il linguaggio di programmazione Python, in particolare ci si è serviti delle librerie:

- **Pandas & numpy**, per poter capire come i dati sono organizzati e per poterli manipolare;
- **seaborn & matplotlib**, per creare grafici adatti a rappresentare lo stato delle informazioni.

Di seguito, i dettagli dei dataset utilizzati.

#### 3.1 dataset PoliceKillingsUS

Il primo dataset da cui si è partiti è reperibile a questo link <sup>[2]</sup>, e si presenta in formato .csv.

Come si evince dalla seguente immagine, il dataset contiene 2535 record, descritti da 14 attributi diversi, di cui si riporta anche il tipo, il conteggio dei valori e il conteggio dei valori nulli:

```
RangeIndex: 2535 entries, 0 to 2534
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                    2535 non-null   int64
1   name                                 2535 non-null   object
2   date                                 2535 non-null   object
3   manner_of_death                     2535 non-null   object
4   armed                               2526 non-null   object
5   age                                  2458 non-null   float64
6   gender                               2535 non-null   object
7   race                                 2340 non-null   object
8   city                                 2535 non-null   object
9   state                               2535 non-null   object
10  signs_of_mental_illness              2535 non-null   bool
11  threat_level                         2535 non-null   object
12  flee                                 2470 non-null   object
13  body_camera                          2535 non-null   bool
```

```
Null values count:
id                0
name              0
date              0
manner_of_death  0
armed             9
age              77
gender            0
race             195
city              0
state             0
signs_of_mental_illness  0
threat_level      0
flee              65
body_camera       0
```

---

<sup>2</sup> <https://www.kaggle.com/kwulum/fatal-police-shootings-in-the-us?select=PoliceKillingsUS.csv>

Cominciamo ora ad esaminare nel dettaglio tutti gli attributi:

- **id** contiene un numero intero progressivo, necessario per identificare univocamente i data objects presenti nel dataset. È un attributo di tipo **numerico discreto**. Non contiene valori nulli.
- **name** contiene i nomi delle vittime, è quindi un attributo di tipo **categorico nominale**. Non contiene valori nulli.
- **date** contiene la data in cui è avvenuto l'intervento della polizia, è quindi un attributo di tipo **nominale**. Non contiene valori nulli. Visto che viene riportata come una variabile di tipo *object*, per poter effettivamente verificare quale sia l'intervallo temporale che viene coperto in questo dataset, cambiamo il tipo di dato in *datetime64[ns]* e provvediamo a verificare quali siano la prima data utile e l'ultima:

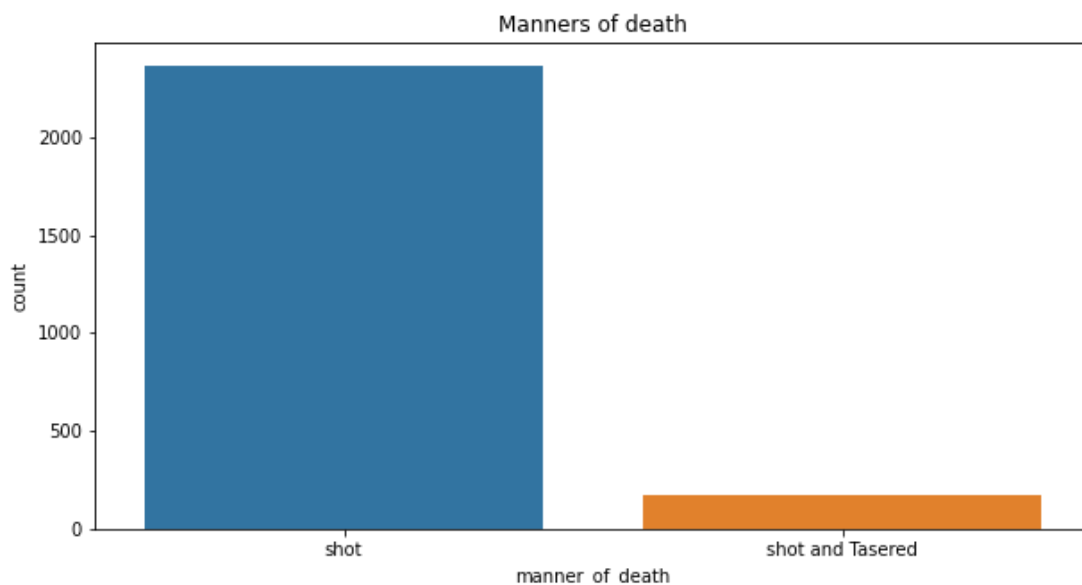
```
min date:
2015-01-03 00:00:00

max date:
2017-12-07 00:00:00
```

Come si evince, la prima data riportata, cronologicamente parlando, è il 03/01/2015 mentre l'ultima risale al 07/12/2017. Pertanto, conserviamo questi dati per le successive integrazioni di altri dataset, tenendo conto dell'intervallo di tempo appena rinvenuto.

- **manner\_of\_death** contiene le cause della morte per una determinata vittima, è quindi un attributo di tipo **categorico nominale**. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

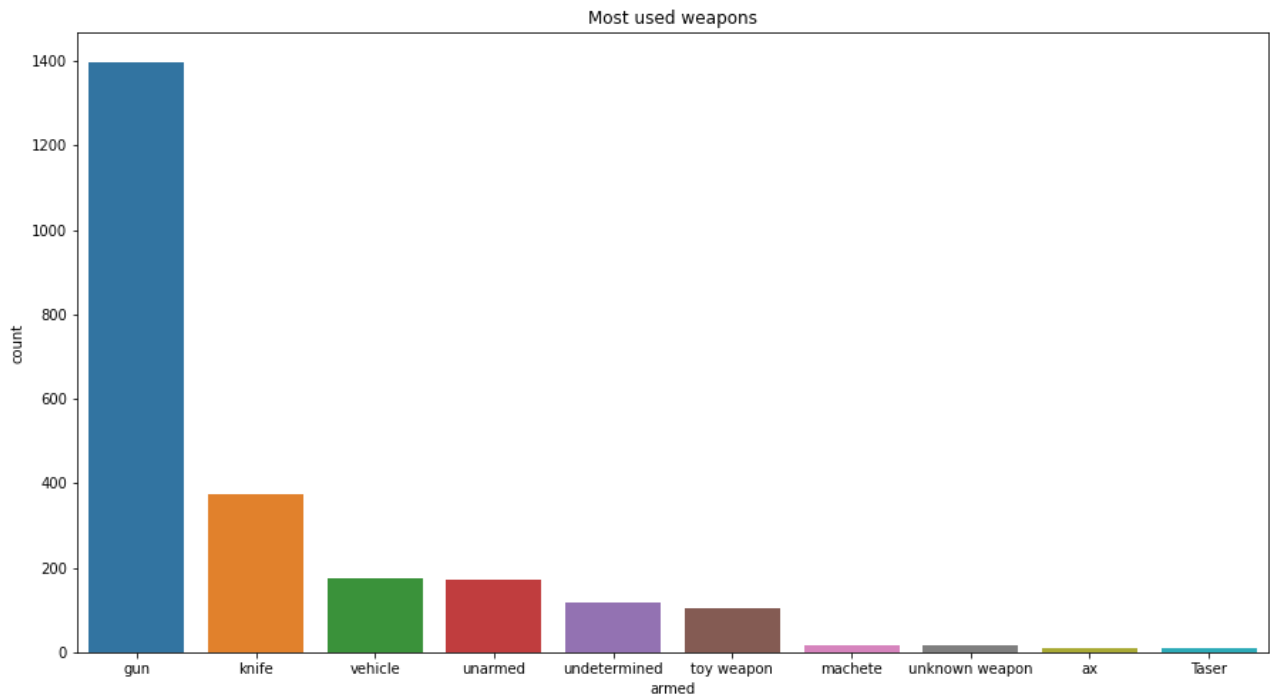
shot	2363
shot and Tasered	172



- **armed** contiene il tipo di arma di cui la vittima era munita, è quindi un attributo di tipo **categorico nominale**. Questa colonna contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, in formato tabellare, mentre in quello grafico si è preferito dare visibilità alle armi più utilizzate:

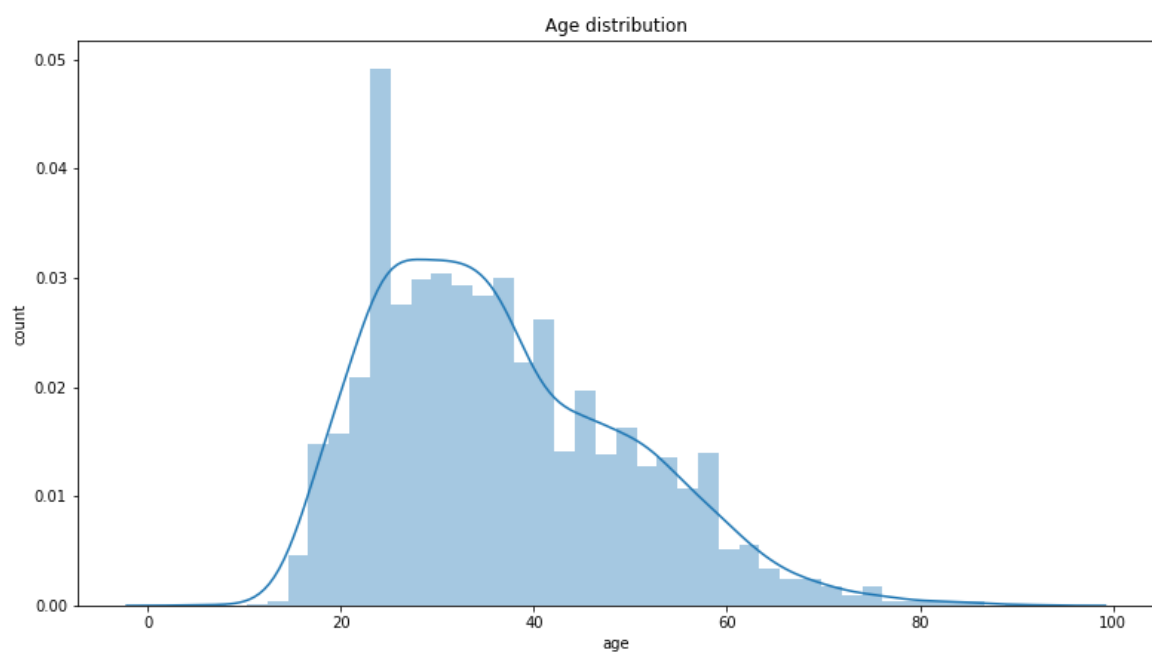
gun	1398	crossbow	6
knife	373	blunt object	5
vehicle	177	gun and knife	5
unarmed	171	screwdriver	5
undetermined	117	scissors	3
toy weapon	104	metal stick	3
unknown weapon	18	guns and explosives	3
machete	18	rock	3
ax	9	sharp object	2
Taser	9	metal pole	2
sword	8	piece of wood	2
baseball bat	8	meat cleaver	2
box cutter	7	chain saw	2
hammer	7	shovel	2
metal pipe	7	pipe	2
hatchet	6	baton	2

metal object	2	crowbar	1
beer bottle	2	machete and gun	1
hatchet and gun	2	spear	1
brick	2	motorcycle	1
baseball bat and bottle	1	pole	1
baseball bat and fireplace poker	1	nail gun	1
bayonet	1	stapler	1
carjack	1	fireworks	1
straight edge razor	1	metal rake	1
garden tool	1	tire iron	1
contractor's level	1	pole and knife	1
metal hand tool	1	pen	1
pitchfork	1	glass shard	1
hand torch	1	bean-bag gun	1
chain	1	pick-axe	1
flashlight	1	air conditioner	1
oar	1	flagpole	1
lawn mower blade	1	cordless drill	1



- **age** contiene l'età della vittima, è quindi un attributo **numerico**, in questo caso di tipo **continuo**. Questa colonna contiene valori nulli. Si riportano di seguito i valori di MINIMO, MASSIMO e MEDIA presenti nella colonna, e anche la distribuzione dell'età:

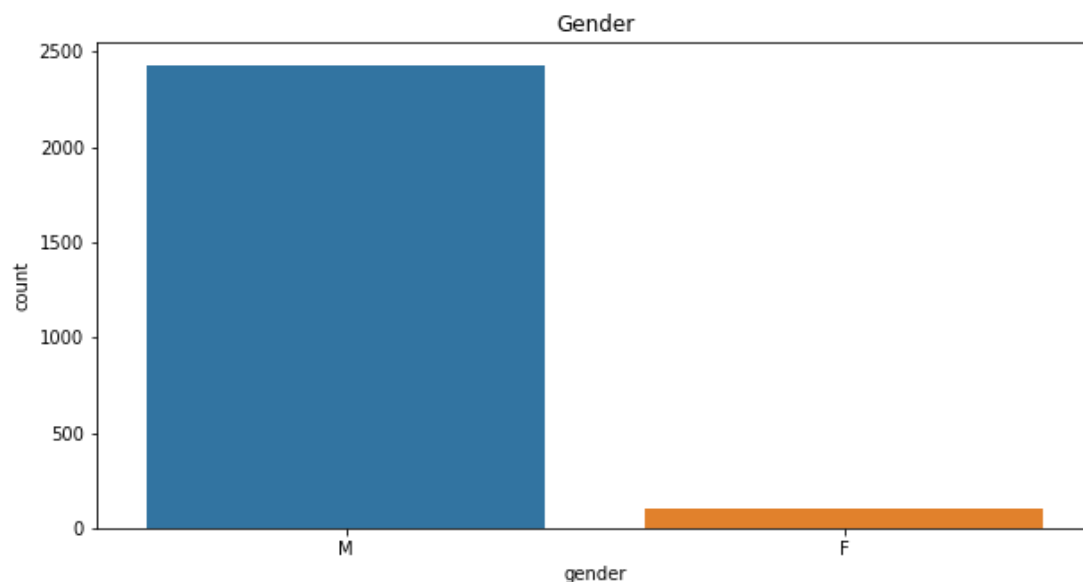
min: 6 max: 91 mean: 36.61





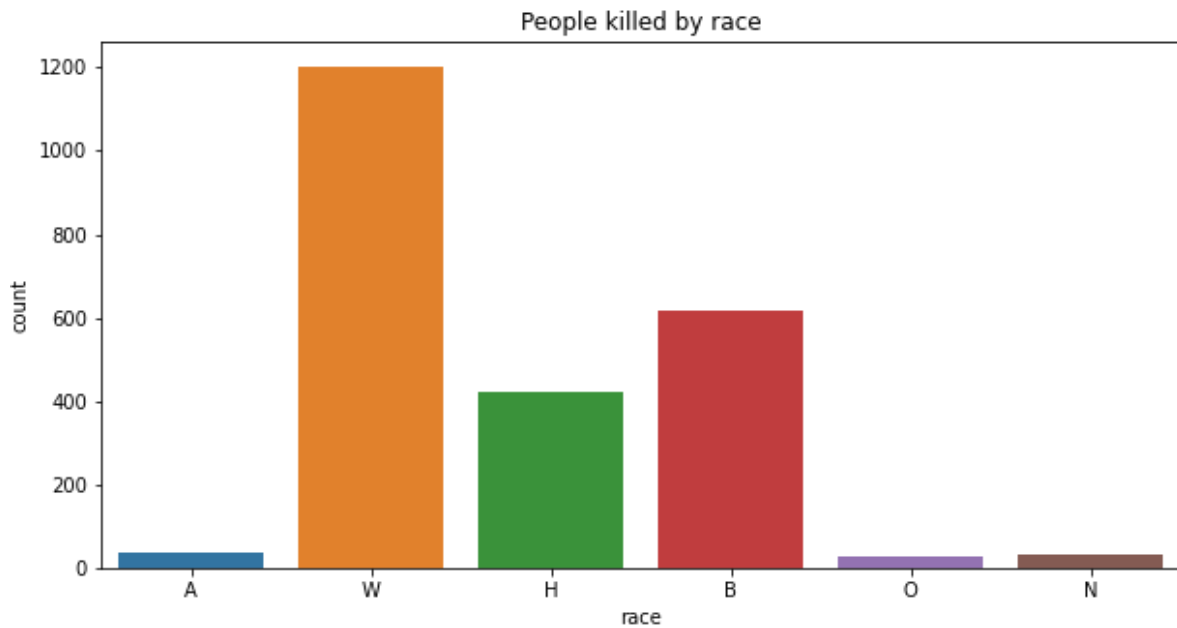
- **gender** contiene il sesso della vittima, è quindi un attributo **categorico nominale**. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

M	2428
F	107



- **race** contiene l'etnia di origine della vittima, è quindi un attributo **categorico nominale**. Contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

W	1201
B	618
H	423
A	39
N	31
O	28



dove:

- W indica WHITE;
- B indica BLACK;
- H indica HISPANIC;
- A indica ASIAN;
- N indica NATIVE-AMERICAN;
- O indica OTHER.

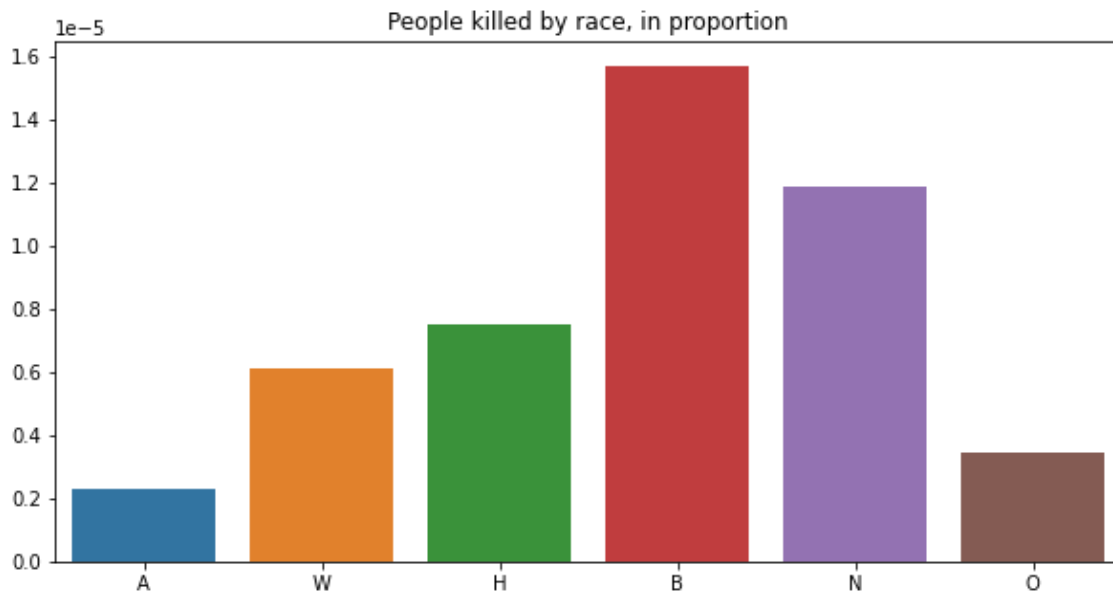
Come si deduce dal grafico, la maggior parte delle vittime è di etnia d'origine bianca, ma questo dato può essere fuorviante, visto che la maggior parte della popolazione americana è proprio costituita da individui appartenenti a questa etnia.

Per poter avere un dato più rappresentativo, si è pensato di ridisegnare il grafico, proporzionando i dati delle vittime all'effettivo numero di individui appartenenti a ciascuna etnia presenti negli Stati Uniti. Sono quindi stati recuperati i seguenti dati, da questo link <sup>[3]</sup>:

Hispanic or Latino (of any race)	56,510,571	17.6%
Mexican	35,709,528	11.1%
Puerto Rican	5,418,521	1.7%
Cuban	2,158,962	0.7%
Other Hispanic or Latino	13,223,560	4.1%
Not Hispanic or Latino	264,493,836	82.4%
White (non-Hispanic)	197,277,789	61.5%
Black or African American (non-Hispanic)	39,445,495	12.3%
American Indian and Alaska Native (non-Hispanic)	2,098,763	0.7%
Asian (non-Hispanic)	16,989,540	5.3%
Native Hawaiian and Other Pacific Islander (non-Hispanic)	515,522	0.2%
Some other race (non-Hispanic)	715,432	0.2%
Two or more races	7,451,295	2.3%

<sup>3</sup> [https://en.wikipedia.org/wiki/Demographics\\_of\\_the\\_United\\_States#Race](https://en.wikipedia.org/wiki/Demographics_of_the_United_States#Race)

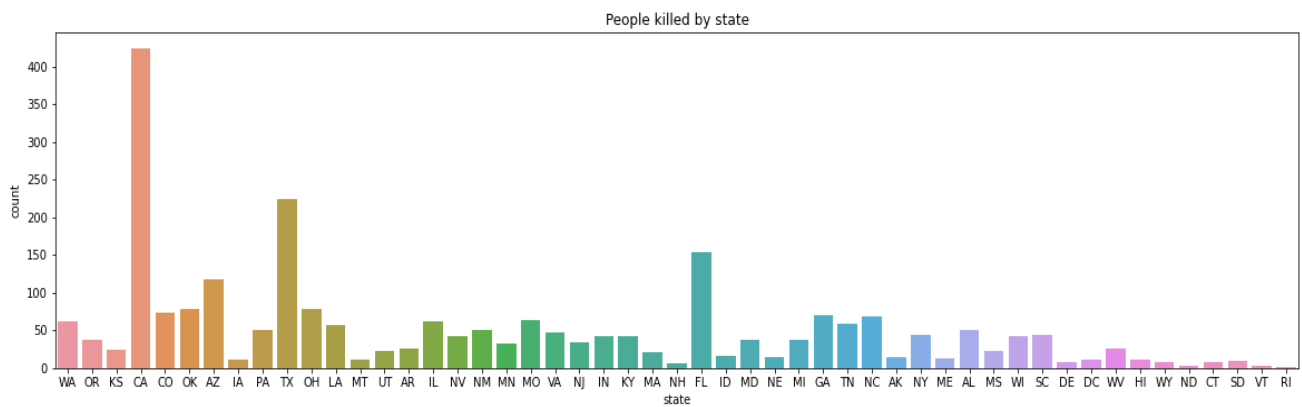
Le frequenze di ogni etnia, presenti nel grafico precedente, sono state rapportate ai dati appena riportati, da cui il grafico seguente:



Si evince chiaramente come il numero di vittime per mano della polizia statunitense sia più orientato verso gli individui di etnia afroamericana e nativo-americana, andando quindi a dare più veridicità all'analisi proposta.

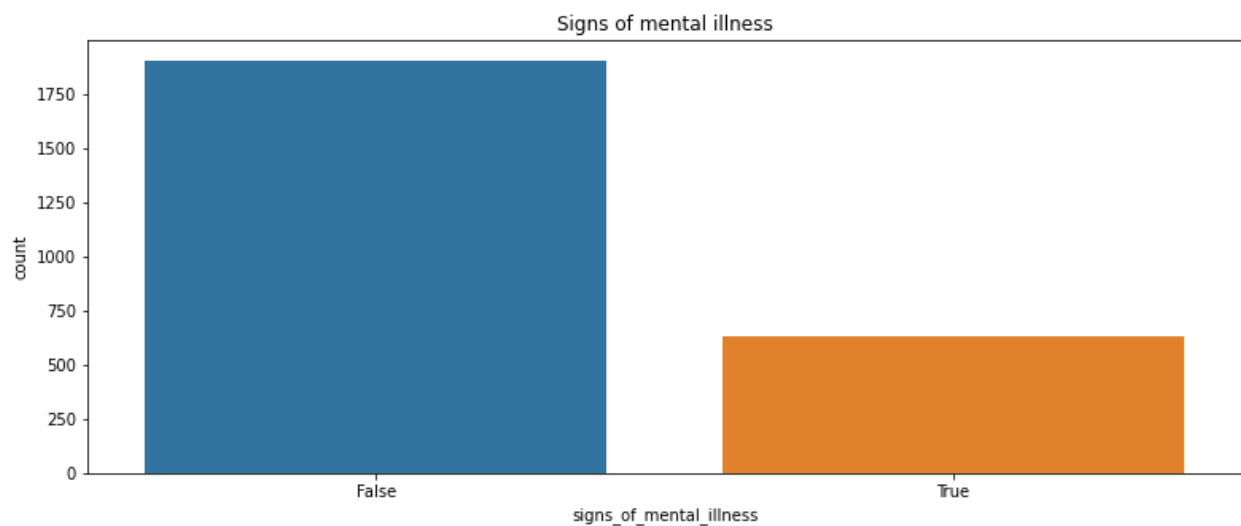
- **city** contiene la città americana in cui è avvenuto l'intervento della polizia, è quindi un attributo **categorico nominale**. Non contiene valori nulli.
- **state** contiene *il codice abbreviato USPS* dello stato americano in cui è avvenuto l'intervento della polizia, è quindi un attributo **categorico nominale**. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

CA	424	VA	47	MA	22
TX	225	NY	45	ID	17
FL	154	SC	44	AK	15
AZ	118	WI	43	NE	15
OH	79	IN	43	ME	13
OK	78	KY	43	IA	12
CO	74	NV	42	HI	11
GA	70	OR	38	DC	11
NC	69	MD	38	MT	11
MO	64	MI	37	SD	10
WA	62	NJ	35	CT	9
IL	62	MN	32	WY	8
TN	59	WV	27	DE	8
LA	57	AR	26	NH	7
PA	51	KS	24	ND	4
NM	51	UT	23	VT	3
AL	50	MS	23	RI	2



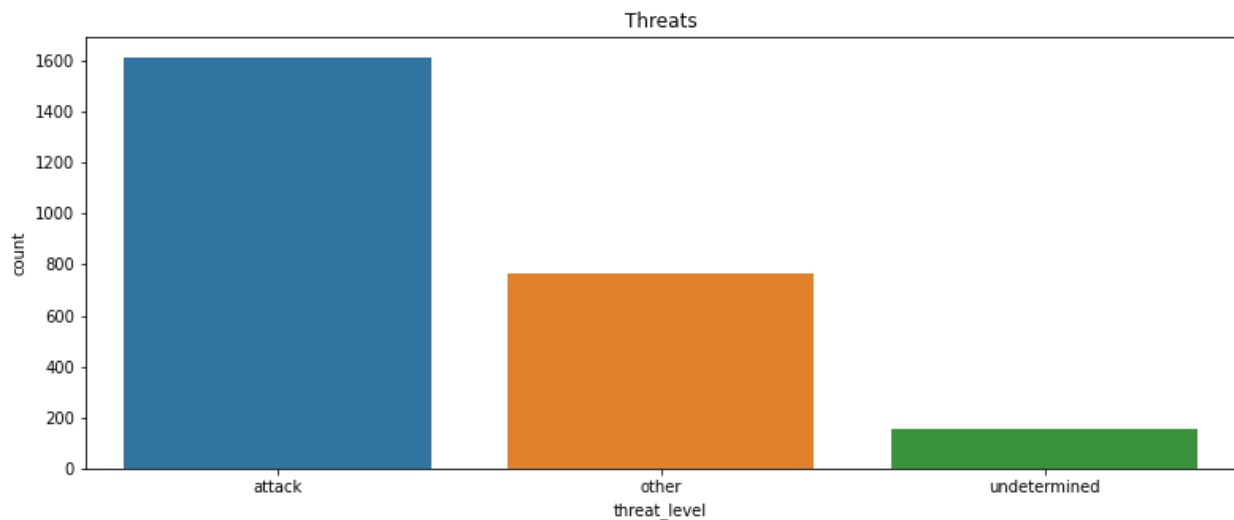
- **signs\_of\_mental\_illness** contiene un valore booleano, indicante se la vittima è stata indicata come mentalmente instabile o meno. È quindi un attributo **categorico nominale**. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

False	1902
True	633



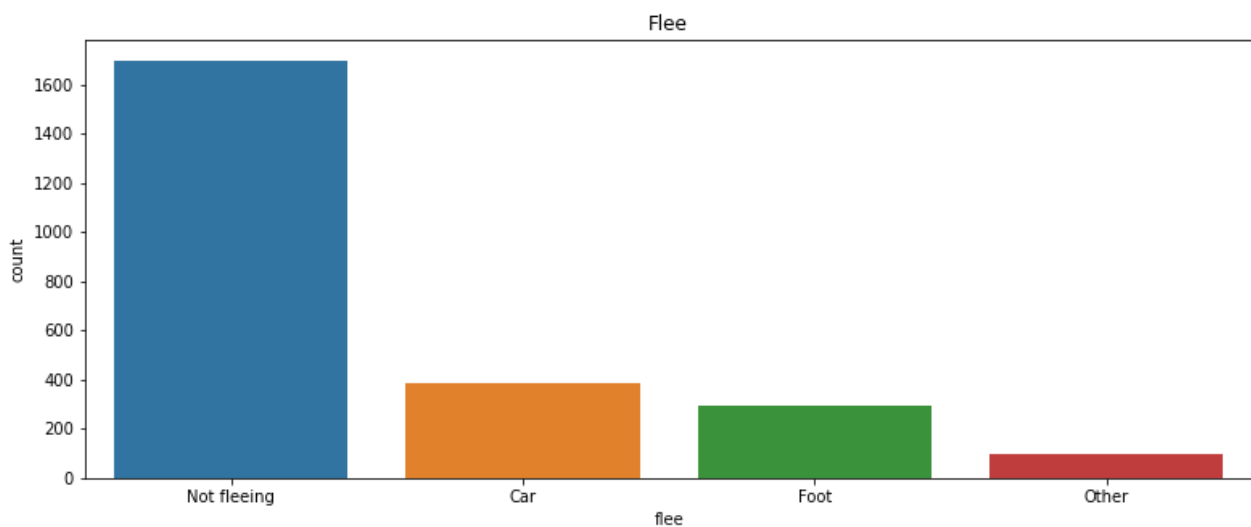
- **threat\_level** contiene il livello di pericolosità della vittima, indicato dall'agente che ha commesso l'intervento. È un attributo di tipo **categorico nominale**. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

<b>attack</b>	<b>1611</b>
<b>other</b>	<b>766</b>
<b>undetermined</b>	<b>158</b>



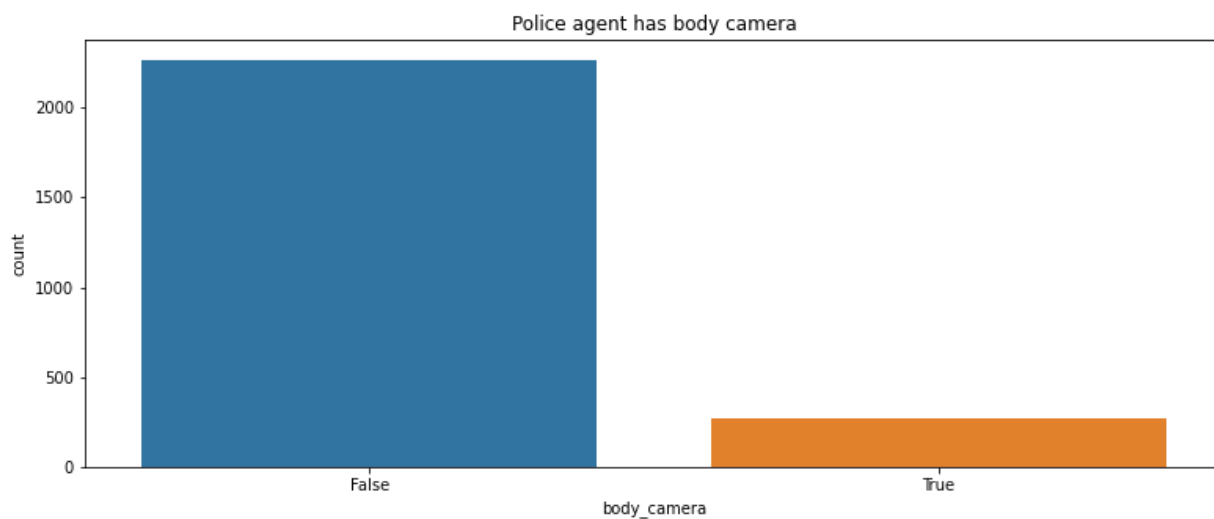
- **flee** contiene il valore indicante se la vittima ha tentato la fuga, e se sì come ha cercato di farlo. È un attributo di tipo **categorico nominale**. Contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

<b>Not fleeing</b>	<b>1695</b>
<b>Car</b>	<b>385</b>
<b>Foot</b>	<b>294</b>
<b>Other</b>	<b>96</b>



- **body\_camera** contiene un valore booleano, indicante se l'agente di polizia che ha commesso l'intervento era dotato di una telecamera. È un attributo di tipo **categorico** nominale. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, sia in formato tabellare che grafico:

False	2264
True	271



### 3.2 dataset stati americani

I dataset nominati in questo paragrafo sono reperibili a questo link <sup>[4]</sup>.

Nella sezione del sito “*Incorporated Places: 2010 to 2019*” sono presenti ben 50 dataset, uno per stato americano. Ogni dataset contiene le informazioni relative alle città che ne fanno parte e al numero di abitanti presenti in ognuna di esse.

I dataset sono in formato *Excel*. Si riporta di seguito la struttura tipo dei suddetti, prendendo come riferimento i primi record presenti nel dataset dello stato dell’*Alabama*:

Annual Estimates of the Resident Population for Incorporated Places in Alabama: April 1, 2010 to July 1, 2019												
Geographic Area	april 1, 2010		Population Estimate (as of July 1)									
	Census	Estimates Base	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Abbeville city, Alabama	2.688	2.705	2.699	2.694	2.643	2.628	2.608	2.600	2.584	2.575	2.571	2.560
Adamsville city, Alabama	4.522	4.506	4.500	4.493	4.471	4.449	4.420	4.390	4.356	4.327	4.308	4.281
Addison town, Alabama	758	754	751	750	743	742	739	734	731	726	723	718
Akron town, Alabama	356	356	355	347	347	343	338	339	333	332	331	328
Alabaster city, Alabama	30.352	31.112	31.209	31.375	31.684	31.980	32.182	32.772	33.017	33.275	33.413	33.487

Come si evince dall’immagine riportata, gli attributi più rilevanti si dividono in due categorie:

- da una parte abbiamo **Geographic Area**, contenente i nomi completi delle città, affiancati dal nome intero dello stato americano di cui fanno parte.
- dall’altra abbiamo una serie di attributi di tipo **numerico**, che indicano tutti il numero di abitanti presenti nella città considerata, nel corso degli anni (a partire dal 2010 fino al 2019).

<sup>4</sup> <https://www.census.gov/data/tables/time-series/demo/popest/2010s-total-cities-and-towns.html#ds>

### 3.3 dataset PercentOver25CompletedHighSchool

Il secondo dataset utilizzato è reperibile a questo link <sup>[5]</sup>, e si presenta in formato .csv.

Come si evince dalla seguente immagine, il dataset contiene 29329 record, descritti da 3 attributi diversi, di cui si riporta anche il tipo, il conteggio dei valori e il conteggio dei valori nulli:

```
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Geographic Area      29329 non-null  object
1   City                 29329 non-null  object
2   percent_completed_hs 29329 non-null  object
```

```
Null values count:
Geographic Area      0
City                  0
percent_completed_hs 0
```

Cominciamo ora ad esaminare nel dettaglio tutti gli attributi:

- **Geographic Area** contiene *il codice abbreviato USPS* dello stato americano in cui risiede la città di cui si conosce la percentuale di persone diplomate, è quindi un attributo **categorico nominale**. Non contiene valori nulli. Si riportano di seguito i valori presenti all'interno della colonna, con il loro conteggio, in formato tabellare:

PA	1762	MI	692	CO	458	MA	246
TX	1747	IN	680	AZ	451	ID	227
CA	1522	KS	672	NM	443	WY	204
IL	1368	GA	627	TN	430	HI	151
OH	1215	WA	627	WV	405	CT	144
NY	1196	VA	593	ND	401	NV	131
MO	1033	AL	585	SC	396	ME	130
IA	1008	NE	580	SD	391	VT	121
FL	918	NJ	545	OR	377	NH	97
MN	903	AR	541	MT	365	DE	77
WI	777	KY	540	MS	362	RI	36
OK	743	MD	518	AK	355	DC	1
NC	739	LA	474	UT	325		

<sup>5</sup> <https://www.kaggle.com/kwullum/fatal-police-shootings-in-the-us?select=PercentOver25CompletedHighSchool.csv>



- **City** contiene tutte le città di cui si conosce la percentuale di persone diplomate, è quindi un attributo **categorico nominale**. Per questioni di leggibilità, si è preferito non inserire alcun dato relativo al conteggio di questo attributo, in quanto i data objects presenti sono troppo numerosi da poter essere visualizzati.
- **percent\_completed\_hs** contiene il valore in percentuale delle persone diplomate per una specifica città. Il fatto che nella prima immagine venga indicato come tipo dell'attributo "object", ci lascia pensare che ci siano dei valori fuori il dominio numerico. Di regola, questo attributo dovrebbe essere di tipo **numerico continuo**. Proviamo a vedere alcuni dei valori presenti, per verificare discordanze:

100	1301
-	197
91.7	170
92.9	169
92.5	168

ne vien fuori che ben 197 data objects hanno il valore "-". Lo terremo presente per la fase di **data preparation**.

### 3.4 dataset PercentagePeopleBelowPovertyLevel

Il terzo dataset utilizzato è reperibile a questo link <sup>[6]</sup>, e si presenta in formato .csv.

Come si evince dalla seguente immagine, il dataset contiene 29329 record, descritti da 3 attributi diversi, di cui si riporta anche il tipo, il conteggio dei valori e il conteggio dei valori nulli:

```
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Geographic Area  29329 non-null  object
1   City            29329 non-null  object
2   poverty_rate    29329 non-null  object
```

```
Null values count:
Geographic Area    0
City               0
poverty_rate       0
```

Le considerazioni effettuate per il dataset precedente ([PercentOver25CompletedHighSchool](#)), relativamente agli attributi indicati dagli indici 0 e 1 (*Geographic Area & City*), sono le stesse che si farebbero per questo dataset, in quanto attributi e valori sono i medesimi. L'unico elemento differente è l'ultimo attributo, indicato dall'indice 2 **poverty\_rate**, il quale ha, visibilmente, un nome diverso rispetto alla colonna indicata dall'indice 2 del dataset precedente (*percent\_completed\_hs*), ma le considerazioni sui suoi valori sono identiche al caso già esaminato.

Pertanto, si decide di procedere all'analisi del dataset successivo.

<sup>6</sup> <https://www.kaggle.com/kwulum/fatal-police-shootings-in-the-us?select=PercentagePeopleBelowPovertyLevel.csv>

### 3.5 dataset MedianHouseholdIncome2015

Il quarto dataset utilizzato è reperibile a questo link <sup>[7]</sup>, e si presenta in formato .csv.

Come si evince dalla seguente immagine, il dataset contiene 29322 record, descritti da 3 attributi diversi, di cui si riporta anche il tipo, il conteggio dei valori e il conteggio dei valori nulli:

```
RangeIndex: 29322 entries, 0 to 29321
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Geographic Area  29322 non-null  object
1   City            29322 non-null  object
2   Median Income   29321 non-null  object
```

```
Null values count:
Geographic Area    0
City               0
Median Income      1
```

Le considerazioni effettuate per i dataset precedenti ([PercentOver25CompletedHighSchool](#) & [PercentagePeopleBelowPovertyLevel](#)), relativamente agli attributi indicati dagli indici 0 e 1 (*Geographic Area & City*), sono le stesse che si farebbero per questo dataset, in quanto attributi e valori sono i medesimi. L'unico elemento differente è l'ultimo attributo, indicato dall'indice 2 **Median Income**, il quale ha, visibilmente, un nome diverso rispetto alla colonna indicata dall'indice 2 dei dataset precedenti (*percent\_completed\_hs & poverty\_rate*), ma le considerazioni sui suoi valori sono identiche al caso già esaminato, eccezion fatta del possibile tipo finale dell'attributo, che si può prevedere essere **numerico discreto**.

Pertanto, si decide di procedere all'analisi del dataset successivo.

---

<sup>7</sup> <https://www.kaggle.com/kwulum/fatal-police-shootings-in-the-us?select=MedianHouseholdIncome2015.csv>

### 3.6 dataset ShareRaceByCity

Il quarto dataset utilizzato è reperibile a questo link <sup>[8]</sup>, e si presenta in formato .csv.

Come si evince dalla seguente immagine, il dataset contiene 29268 record, descritti da 7 attributi diversi, di cui si riporta anche il tipo, il conteggio dei valori e il conteggio dei valori nulli:

```
RangeIndex: 29268 entries, 0 to 29267
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Geographic area       29268 non-null  object
1   City                  29268 non-null  object
2   share_white           29268 non-null  object
3   share_black           29268 non-null  object
4   share_native_american 29268 non-null  object
5   share_asian           29268 non-null  object
6   share_hispanic        29268 non-null  object
```

```
Null values count:
Geographic area    0
City               0
share_white        0
share_black        0
share_native_american 0
share_asian        0
share_hispanic     0
```

Le considerazioni effettuate per i dataset precedenti ([PercentOver25CompletedHighSchool](#) & [PercentagePeopleBelowPovertyLevel](#) & [MedianHouseholdIncome2015](#)), relativamente agli attributi indicati dagli indici 0 e 1 (*Geographic Area* & *City*), sono le stesse che si farebbero per questo dataset, in quanto attributi e valori sono i medesimi.

Analizziamo più da vicino i restanti:

- **share\_white** indica la percentuale di individui di etnia d'origine *bianca* presenti nella città corrispondente. Il fatto che nella prima immagine riportata venga indicato come tipo dell'attributo "object", ci lascia pensare che ci siano dei valori fuori il dominio numerico. Di regola, questo attributo dovrebbe essere di tipo **numerico continuo**.
- **share\_black** indica la percentuale di individui di etnia d'origine *afroamericana* presenti nella città corrispondente. Le considerazioni effettuate per l'attributo precedente sono le medesime.
- **share\_native\_american** indica la percentuale di individui di etnia d'origine *nativo-americana* presenti nella città corrispondente. Le considerazioni effettuate per l'attributo precedente sono le medesime.
- **share\_asian** indica la percentuale di individui di etnia d'origine *asiatica* presenti nella città corrispondente. Le considerazioni effettuate per l'attributo precedente sono le medesime.
- **share\_hispanic** indica la percentuale di individui di etnia d'origine *ispanica* presenti nella città corrispondente. Le considerazioni effettuate per l'attributo precedente sono le medesime.

<sup>8</sup> <https://www.kaggle.com/kwulum/fatal-police-shootings-in-the-us?select=ShareRaceByCity.csv>

## 4. Data Preparation

Alla luce delle esplorazioni fatte, i dataset proposti risultano in forma grezza e incompleta e presentano delle anomalie, pertanto al momento risulta impossibile avviare un'analisi su *Weka*, lo strumento scelto per l'attività di *Data mining*. Si comincia quindi l'attività di *Data Preparation*, che verrà effettuata in maniera incrementale, dataset per dataset, andando di volta in volta a selezionare i dati, pulirli, formattarli e integrarli. Si utilizza, come già anticipato nella fase di [Data Understanding](#), il linguaggio Python, in particolare le librerie **pandas**, **numpy**. Per i dataset degli *stati americani*, invece, si farà riferimento allo strumento di *Power Query* di *Microsoft Excel*.

### 4.1 dataset PoliceKillingsUS

Ai fini dell'analisi proposta, si decide di eliminare alcuni attributi, ritenuti poco utili. Gli attributi da eliminare sono: **id**, **name**, **manner\_of\_death** e **date**, di cui però si conserva l'intervallo temporale identificato precedentemente (03/01/2015 – 07/12/2017).

#### 4.1.1 Caricamento in memoria del dataset

Come prima cosa, si carica in memoria il dataset tramite il seguente comando:

```
police_killings = pd.read_csv(r'C:\Users\rinos\OneDrive\Desktop\ITPS\TERZO ANNO\SECONDO SEMESTRE\Data mining\progetto\police shootings\PoliceKillingsUS.csv')
```

#### 4.1.2 Eliminazione attributi

si procede all'eliminazione degli attributi indicati:

```
police_killings.drop(['id', 'date', 'name', 'manner_of_death'], axis=1, inplace=True)
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

	armed	age	gender	race	city	state	signs_of_mental_illness	threat_level	flee	body_camera
	gun	53.0	M	A	Shelton	WA	True	attack	Not fleeing	False
	gun	47.0	M	W	Aloha	OR	False	attack	Not fleeing	False
	unarmed	23.0	M	H	Wichita	KS	False	other	Not fleeing	False
	toy weapon	32.0	M	W	San Francisco	CA	True	attack	Not fleeing	False
	nail gun	39.0	M	H	Evans	CO	False	attack	Not fleeing	False

### 4.1.3 Attributo age

Si procede successivamente ad esaminare l'attributo *age*, il quale, come visto, aveva dei valori mancanti. Si decide di rimpiazzarli con la media di tutti i record presenti nel dataset, dopodiché ne si cambia il tipo di dato, da *float* a *int*.

```
police_killings['age'].fillna(value=police_killings['age'].mean(), inplace=True)
police_killings['age'] = police_killings['age'].astype(int)
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

	armed	age	gender	race	city	state	signs_of_mental_illness	threat_level	flee	body_camera
	gun	53	M	A	Shelton	WA	True	attack	Not fleeing	False
	gun	47	M	W	Aloha	OR	False	attack	Not fleeing	False
	unarmed	23	M	H	Wichita	KS	False	other	Not fleeing	False
	toy weapon	32	M	W	San Francisco	CA	True	attack	Not fleeing	False
	nail gun	39	M	H	Evans	CO	False	attack	Not fleeing	False

### 4.1.4 Attributo city

Considerando che la maggior parte dei dataset ha un riferimento ad una città, si decide di formattare questo attributo nella forma "*nome\_città, codice\_stato*", in modo da facilitare i successivi *merge* con gli altri dataset.

Si decide quindi di eliminare dai valori dell'attributo *city* tutte le parole, o sottostringhe, che potrebbero compromettere l'attività appena citata. Le parole da eliminare sono "city, City, CDP, town, Town, village, Village".

```
police_killings['city'].replace(['city', 'City', 'CDP', 'town', 'Town', 'village', 'Village'], '', regex=True, inplace=True)
```

Successivamente, si formattano i valori dell'attributo, in modo da rispettare la forma indicata "*nome\_città, codice\_stato*":

```
police_killings['city'] = police_killings['city'] + ', ' + police_killings['state']
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

	armed	age	gender	race	city	state	signs_of_mental_illness	threat_level	flee	body_camera
	gun	53	M	A	Shelton, WA	WA	True	attack	Not fleeing	False
	gun	47	M	W	Aloha, OR	OR	False	attack	Not fleeing	False
	unarmed	23	M	H	Wichita, KS	KS	False	other	Not fleeing	False
	toy weapon	32	M	W	San Francisco, CA	CA	True	attack	Not fleeing	False
	nail gun	39	M	H	Evans, CO	CO	False	attack	Not fleeing	False

#### 4.1.5 Attributo state

Dunque, si può eliminare l'attributo *state*, visto che è stato inglobato nell'attributo *city*:

```
police_killings.drop('state', axis=1, inplace=True)
```

e ne si visualizza il risultato:

```
police_killings.head()
```

	armed	age	gender	race	city	signs_of_mental_illness	threat_level	flee	body_camera
	gun	53	M	A	Shelton, WA	True	attack	Not fleeing	False
	gun	47	M	W	Aloha, OR	False	attack	Not fleeing	False
	unarmed	23	M	H	Wichita, KS	False	other	Not fleeing	False
	toy weapon	32	M	W	San Francisco, CA	True	attack	Not fleeing	False
	nail gun	39	M	H	Evans, CO	False	attack	Not fleeing	False

#### 4.1.6 Attributo armed

Come visto precedentemente, l'attributo contiene numerosi valori. Per facilitare l'analisi successiva, si decide di cambiare il valore di ciascun record relativamente all'attributo *armed*, il quale potrà assumere solo i valori *Yes* e *No*. Per far questo, si procede a sostituire ogni occorrenza dei valori *unarmed*, *undetermined* e *NaN* (valori nulli), con *No*, mentre tutti gli altri con *Yes*:

```
for i in police_killings.index:
    armed = police_killings.at[i, 'armed']
    if isnull(armed) or armed == 'unarmed' or armed == 'undetermined':
        police_killings.at[i, 'armed'] = 'No'
    else:
        police_killings.at[i, 'armed'] = 'Yes'
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

armed	age	gender	race	city	signs_of_mental_illness	threat_level	flee	body_camera
Yes	53	M	A	Shelton, WA	True	attack	Not fleeing	False
Yes	47	M	W	Aloha, OR	False	attack	Not fleeing	False
No	23	M	H	Wichita, KS	False	other	Not fleeing	False
Yes	32	M	W	San Francisco, CA	True	attack	Not fleeing	False
Yes	39	M	H	Evans, CO	False	attack	Not fleeing	False

Per completezza, si stampano i valori contenuti nella colonna:

```
police_killings['armed'].value_counts().to_frame().style
```

	armed
Yes	2238
No	297

#### 4.1.7 Attributo flee

Per le stesse ragioni esposte nel [paragrafo precedente](#), si decide di rimappare i valori dell'attributo *flee* in *Yes* e *No*. Per far questo, si procede a sostituire ogni occorrenza dei valori *Not fleeing* e *NaN* (valori nulli), con *No*, mentre tutti gli altri con *Yes*:

```
for i in police_killings.index:
    flee = police_killings.at[i, 'flee']
    if isnull(flee) or flee == 'Not fleeing':
        police_killings.at[i, 'flee'] = 'No'
    else:
        police_killings.at[i, 'flee'] = 'Yes'
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

police\_killings.head()

armed	age	gender	race	city	signs_of_mental_illness	threat_level	flee	body_camera
Yes	53	M	A	Shelton, WA	True	attack	No	False
Yes	47	M	W	Aloha, OR	False	attack	No	False
No	23	M	H	Wichita, KS	False	other	No	False
Yes	32	M	W	San Francisco, CA	True	attack	No	False
Yes	39	M	H	Evans, CO	False	attack	No	False

Per completezza, si stampano i valori contenuti nella colonna:

```
police_killings['flee'].value_counts().to_frame().style
```

	flee
No	1760
Yes	775

#### 4.1.8 Attributo threat\_level

Per le stesse ragioni esposte nel [paragrafo precedente](#), si decide di rimappare i valori dell'attributo *threat\_level* in *Yes* e *No*. Per far questo, si procede a sostituire ogni occorrenza dei valori *undetermined* e *NaN* (valori nulli), con *No*, mentre tutti gli altri con *Yes*. Prima di questo, si decide di rinominare la colonna in *threat*:

```
police_killings = police_killings.rename(columns={'threat_level': 'threat'})

for i in police_killings.index:
    threat = police_killings.at[i, 'threat']
    if isnull(threat) or threat == 'undetermined':
        police_killings.at[i, 'threat'] = 'No'
    else:
        police_killings.at[i, 'threat'] = 'Yes'
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

armed	age	gender	race	city	signs_of_mental_illness	threat	flee	body_camera
Yes	53	M	A	Shelton, WA	True	Yes	No	False
Yes	47	M	W	Aloha, OR	False	Yes	No	False
No	23	M	H	Wichita, KS	False	Yes	No	False
Yes	32	M	W	San Francisco, CA	True	Yes	No	False
Yes	39	M	H	Evans, CO	False	Yes	No	False

Per completezza, si stampano i valori contenuti nella colonna:

```
police_killings['threat'].value_counts().to_frame().style
```

	threat
Yes	2377
No	158



#### 4.1.9 Attributo race

Questo attributo sarà quello *di classe* per la nostra analisi. Si decide quindi di rinominarlo in *is\_black* e di rimappare i suoi valori in *Yes* e *No*. Prima della rimappatura, occorre gestire i valori nulli/mancanti che vengono prima sostituiti con un valore randomico, scelto tra i possibili valori che l'attributo *race* può assumere (ovvero *A, W, H, B, N, O*). Per far questo, ci serviamo della funzione *random\_race* così definita:

```
def random_race():
    races_list = ['A', 'W', 'H', 'B', 'N', 'O']
    random_num = int(random.random() * 10**6)
    return races_list[random_num % 6]
```

Definita la funzione, possiamo procedere:

```
police_killings = police_killings.rename(columns={'race': 'is_black'})
for i in police_killings.index:
    race = police_killings.at[i, 'is_black']
    if isnull(race):
        police_killings.at[i, 'is_black'] = random_race()

    police_killings.at[i, 'is_black'] = 'Yes' if race == 'B' else 'No'
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera
Yes	53	M	No	Shelton, WA	True	Yes	No	False
Yes	47	M	No	Aloha, OR	False	Yes	No	False
No	23	M	No	Wichita, KS	False	Yes	No	False
Yes	32	M	No	San Francisco, CA	True	Yes	No	False
Yes	39	M	No	Evans, CO	False	Yes	No	False

Per completezza, si stampano i valori contenuti nella colonna:

```
police_killings['is_black'].value_counts().to_frame().style
```

is_black	
No	1932
Yes	603

#### 4.1.10 Attributo signs\_of\_mental\_illness

Per le stesse ragioni esposte nel [paragrafo precedente](#), si decide di rimappare i valori dell'attributo *signs\_of\_mental\_illness* in *Yes* e *No*. Per far questo, si procede a sostituire ogni occorrenza dei valori *False* con *No*, mentre tutti gli altri con *Yes*:

```
police_killings['signs_of_mental_illness'] = police_killings['signs_of_mental_illness'].astype(str)
for i in police_killings.index:
    ill = police_killings.at[i, 'signs_of_mental_illness']
    police_killings.at[i, 'signs_of_mental_illness'] = 'Yes' if ill == 'True' else 'No'
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera
Yes	53	M	No	Shelton, WA	Yes	Yes	No	False
Yes	47	M	No	Aloha, OR	No	Yes	No	False
No	23	M	No	Wichita, KS	No	Yes	No	False
Yes	32	M	No	San Francisco, CA	Yes	Yes	No	False
Yes	39	M	No	Evans, CO	No	Yes	No	False

Per completezza, si stampano i valori contenuti nella colonna:

```
police_killings['signs_of_mental_illness'].value_counts().to_frame().style
```

signs_of_mental_illness	
No	1902
Yes	633

#### 4.1.11 Attributo body\_camera

Per le stesse ragioni espresse nel [paragrafo precedente](#), si decide di rimappare i valori dell'attributo *body\_camera* in *Yes* e *No*. Per far questo, si procede a sostituire ogni occorrenza dei valori *False* con *No*, mentre tutti gli altri con *Yes*:

```
police_killings['body_camera'] = police_killings['body_camera'].astype(str)
for i in police_killings.index:
    camera = police_killings.at[i, 'body_camera']
    police_killings.at[i, 'body_camera'] = 'Yes' if camera == 'True' else 'No'
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
police_killings.head()
```

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera
Yes	53	M	No	Shelton, WA	Yes	Yes	No	No
Yes	47	M	No	Aloha, OR	No	Yes	No	No
No	23	M	No	Wichita, KS	No	Yes	No	No
Yes	32	M	No	San Francisco, CA	Yes	Yes	No	No
Yes	39	M	No	Evans, CO	No	Yes	No	No

Per completezza, si stampano i valori contenuti nella colonna:

```
police_killings['body_camera'].value_counts().to_frame().style
```

body_camera	
No	2264
Yes	271

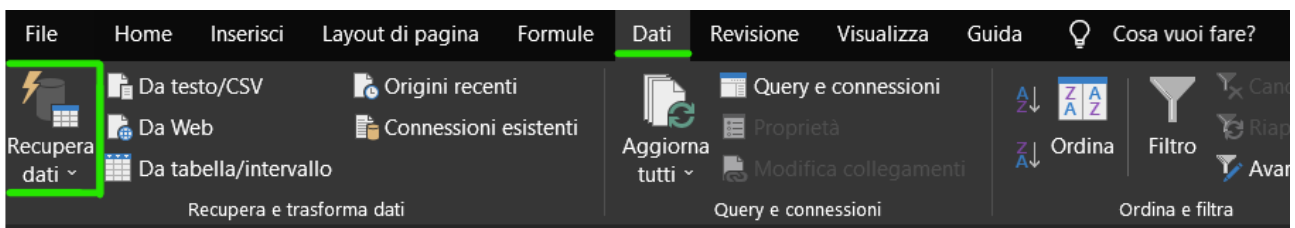
## 4.2 dataset stati americani

Come anticipato all'inizio del [paragrafo di Data Preparation](#), per i dataset degli stati americani si è utilizzato lo strumento *Power Query* di *Microsoft Excel*.

Il lavoro svolto è identico su ogni dataset, in quanto formattati alla stessa maniera e provenienti dalla stessa fonte. Si decide di mostrare quindi soltanto gli interventi effettuati sul primo dataset, quello pertinente lo stato americano dell'*Alabama*.

### 4.2.1 Uso di Power Query di Excel

Si apre un nuovo file *excel* e, dal tab *Dati* presente nella barra degli strumenti in alto, si clicca su *Recupera dati*, e successivamente si carica il dataset *Da File*:



Dalla finestra *Strumento di navigazione* che si è aperta, si clicca su *Trasforma dati*:

Strumento di navigazione

Seleziona più elementi

Opzioni di visualizzazione

SUB-IP-EST2019-ANNRES-01.xlsx [3]

SUB-IP-EST2019-ANNRES-01

\_xlnm.Print\_Area

sub\_annres\_1

SUB-IP-EST2019-ANNRES-01

table with row headers in column A and column headers in rows 3 thr...	Column2	Column3	Column4
Annual Estimates of the Resident Population for Incorporated Places in Al...	null	null	
Geographic Area	01/04/2010	null	Population Estimate (as of July 1)
	null Census	Estimates Base	
Abbeville city, Alabama	2688	2705	
Adamsville city, Alabama	4522	4506	
Addison town, Alabama	758	754	
Akron town, Alabama	356	356	
Alabaster city, Alabama	30352	31112	
Albertville city, Alabama	21160	21209	
Alexander City city, Alabama	14875	14984	
Aliceville city, Alabama	2486	2481	
Allgood town, Alabama	622	622	
Altoona town, Alabama	933	937	
Andalusia city, Alabama	9015	9015	
Anderson town, Alabama	282	281	
Anniston city, Alabama	23106	22987	
Arab city, Alabama	8050	8080	
Ardmore town, Alabama	1194	1194	
Argo town, Alabama	4071	4085	
Ariton town, Alabama	764	762	
Arley town, Alabama	357	357	
Ashford town, Alabama	2148	2149	
Ashland town, Alabama	2037	2039	

Carica Trasforma dati Annulla

Si aprirà quindi *Power Query*.

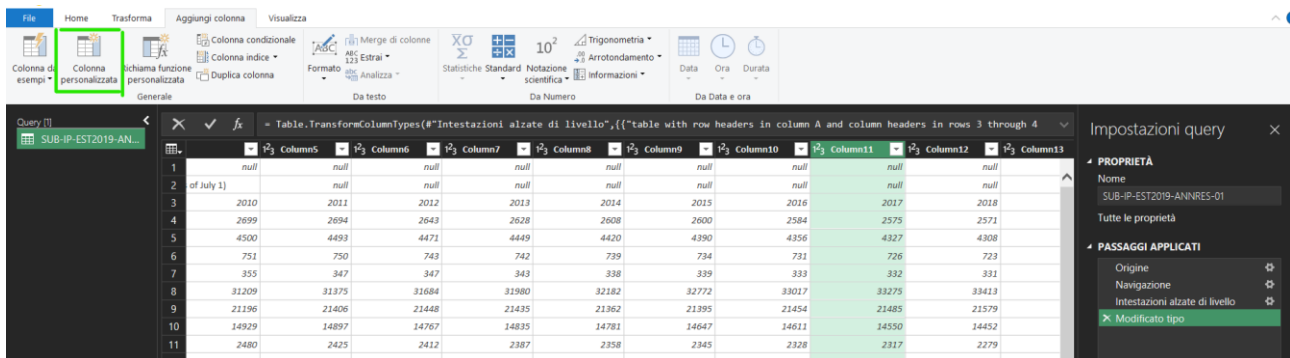
Geographic Area	Census	Estimates Base	Population Estimate (as of July 1)
Abbeville city, Alabama	2688	2705	2694
Adamsville city, Alabama	4522	4506	4493
Addison town, Alabama	758	754	750
Akron town, Alabama	356	356	347
Alabaster city, Alabama	30352	31112	31209
Albertville city, Alabama	21160	21209	21196
Alexander City city, Alabama	14875	14984	14929
Aliceville city, Alabama	2486	2481	2480
Allgood town, Alabama	622	622	622
Altoona town, Alabama	933	937	937
Andalusia city, Alabama	9015	9015	9025
Anderson town, Alabama	282	281	281
Anniston city, Alabama	23106	22987	22932
Arab city, Alabama	8050	8080	8095
Ardmore town, Alabama	1194	1194	1203
Argo town, Alabama	4071	4085	4093
Ariton town, Alabama	764	762	763
Arley town, Alabama	357	357	356
Ashford town, Alabama	2148	2149	2151

A questo punto, si individuano tutte le colonne che hanno come valore il numero di abitanti. Si nota però che il tipo di dato di queste colonne è *Text*, si procede dunque a modificarlo in *Numero decimale*: si selezionano quindi tutte le colonne desiderate > click sul tasto destro del mouse > Modifica tipo > Numero decimale:

Geographic Area	Census	Estimates Base	Population Estimate (as of July 1)
Abbeville city, Alabama	2688	2705	2694
Adamsville city, Alabama	4522	4506	4493
Addison town, Alabama	758	754	750
Akron town, Alabama	356	356	347
Alabaster city, Alabama	30352	31112	31209
Albertville city, Alabama	21160	21209	21196
Alexander City city, Alabama	14875	14984	14929
Aliceville city, Alabama	2486	2481	2480
Allgood town, Alabama	622	622	622
Altoona town, Alabama	933	937	937
Andalusia city, Alabama	9015	9015	9025
Anderson town, Alabama	282	281	281
Anniston city, Alabama	23106	22987	22932
Arab city, Alabama	8050	8080	8095
Ardmore town, Alabama	1194	1194	1203
Argo town, Alabama	4071	4085	4093
Ariton town, Alabama	764	762	763
Arley town, Alabama	357	357	356
Ashford town, Alabama	2148	2149	2151

Si procede ora alla creazione dell'unica colonna che terremo in considerazione per l'analisi, ovvero *population*. Questo nuovo attributo conterrà i valori del numero di abitanti per ogni città, ottenuto facendo una media di tutti i valori presenti nella riga della città selezionata, considerando però l'intervallo temporale che abbiamo individuato nella fase di *Data Understanding*, [in questo paragrafo](#) (03/01/2015 – 07/12/2017), quindi si procederà a fare una media delle colonne 2015, 2016, 2017.

Per farlo, ci si reca nel tab *Aggiungi colonna* presente nella barra degli strumenti in alto, e si clicca su *Colonna personalizzata*:



Nella finestra che appare, si rinomina la nuova colonna in *population* e si applica la formula alle colonne prese in esame, ovvero la 9, 10, 11:

Colonna personalizzata

Aggiunge una colonna che viene calcolata da altre colonne.

Nome nuova colonna

Formula colonna personalizzata

Colonne disponibili  

Column7  
 Column8  
 Column9  
 Column10  
 Column11  
 Column12  
 Column13

<< Inserisci

[Informazioni sulle formule di Power Query](#)

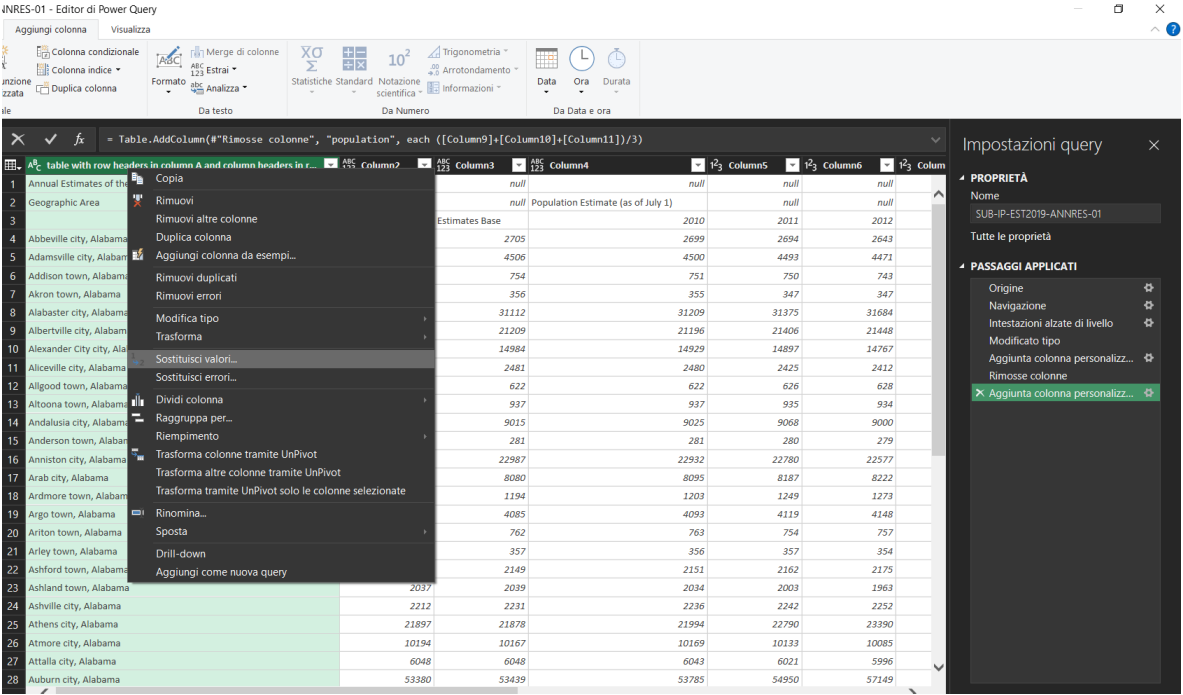
✓ Non sono stati rilevati errori di sintassi.

OK

Annulla

Si è dunque aggiunta la nuova colonna.

Si procede ora a sostituire ad elaborare la prima colonna: essa contiene le città dello stato dell’*Alabama*, nel formato “*nome\_città, nome\_stato*”. Si formattano i valori di questa colonna, andando a sostituire al nome completo dello stato il suo *codice abbreviato USPS*. Per farlo, si clicca con il tasto destro del mouse sull’header della colonna > Sostituisci valori:



Nella finestra che appare, si sostituisce ogni occorrenza di “, *Alabama*” con “, *AL*”:

Sostituisci valori

Sostituire un valore con un altro nelle colonne selezionate.

Valore da trovare

, Alabama

Sostituisci con

, AL

> Opzioni avanzate

OK

Annulla

Arrivati a questo punto, si procede ad eliminare tutte le colonne rimanenti, eccezion fatta per quella appena modificata e quella creata precedentemente (*population*):

	ABC table with row headers in column A and column headers in r...	ABC 123 population
1	Annual Estimates of the Resident Population for Incorporated Places i...	null
2	Geographic Area	null
3	null	2016
4	Abbeville city, AL	2586,333333
5	Adamsville city, AL	4357,666667
6	Addison town, AL	730,333333
7	Akron town, AL	334,666667
8	Alabaster city, AL	33021,33333
9	Albertville city, AL	21444,66667
10	Alexander City city, AL	14602,66667
11	Aliceville city, AL	2330
12	Allgood town, AL	636,333333
13	Altoona town, AL	921
14	Andalusia city, AL	8839,33333
15	Anderson town, AL	272,666667
16	Anniston city, AL	21870,66667
17	Arab city, AL	8266,33333

Si caricano adesso i dati nel file *excel* e si procede alle ultime modifiche in questo software:

- si eliminano le prime e le ultime righe, senza significato;
- si rinomina la prima colonna in *Geographic Area*.

Il risultato finale:

1	Geographic Area	population
2	Abbeville city, AL	2586,333333
3	Adamsville city, AL	4357,666667
4	Addison town, AL	730,333333
5	Akron town, AL	334,666667
6	Alabaster city, AL	33021,33333
7	Albertville city, AL	21444,66667
8	Alexander City city, AL	14602,66667
9	Aliceville city, AL	2330
10	Allgood town, AL	636,333333
11	Altoona town, AL	921
12	Andalusia city, AL	8839,33333
13	Anderson town, AL	272,666667
14	Anniston city, AL	21870,66667
15	Arab city, AL	8266,33333
16	Ardmore town, AL	1376
17	Argo town, AL	4257
18	Ariton town, AL	741,333333
19	Arley town, AL	348,333333
20	Ashford town, AL	2158,66667
21	Ashland town, AL	1937,66667
22	Ashville city, AL	2209,66667

Si salva il file in formato *.csv* per effettuare altre elaborazioni tramite Python.



## 4.2.2 Uso di Python

Inizia la fase finale di preparazione dei dataset americani.

Il lavoro da svolgere su ogni dataset è identico, in quanto formattati ed elaborati alla stessa maniera e provenienti dalla stessa fonte. Si è deciso quindi di costruire una funzione che prende in input il nome dello stato americano, il quale viene manipolato e successivamente viene salvato in un file.

### 4.2.2.1 Elaborazioni sui singoli dataset

Le elaborazioni effettuate sono le seguenti:

- viene rinominato l'attributo *Geographic Area* in *city*, alla stessa maniera di come è stato fatto per i dataset precedenti.

```
dataframe = dataframe.rename(columns={'Geographic Area': 'city'})
```

- Si elimina dai valori dell'attributo *city* tutte le parole, o sottostringhe, che corrispondono a "city, City, CDP, town, Town, village, Village", come è stato fatto per i dataset precedenti.

```
dataframe['city'].replace([' city', ' City', ' CDP', ' town', ' Town', ' village', ' Village'], '', regex=True, inplace=True)
```

- In seguito al lavoro svolto su *Excel*, la colonna *population* ha acquisito il tipo *object*, poiché al suo interno sono stati ritrovati valori anomali ed estranei al tipo *numerico*. Uno dei valori è il segno ',' (virgola) utilizzato per separare le cifre intere da quelle decimali, ed è stato dunque rimpiazzato con il '.' (punto); l'altro valore è il '-', rimpiazzato dal valore *NaN* (nullo).

```
dataframe['population'].replace(',', '.', regex=True, inplace=True)  
dataframe['population'].replace('-', np.NaN, regex=True, inplace=True)
```

- Vengono dunque eliminati tutti i record che hanno un valore nullo.

```
dataframe.dropna()
```

- Infine, si effettua la conversione di tipo, da *object* a *int*, quindi tipo *numerico discreto*.

```
dataframe['population'] = np.round(dataframe['population'].to_numpy(dtype='float32')).astype(int)
```

I passaggi precedenti sono tutti stati utilizzati nella funzione creata *format\_file()*:

```
def format_file(state_name):  
    base_path = 'C:\\Users\\rinos\\OneDrive\\Desktop\\ITPS\\TERZO ANNO\\SECONDO SEMESTRE\\Data  
mining\\progetto\\police shootings\\states\\final\\' + state_name + '.csv'  
    dataframe = pd.read_csv(base_path, delimiter=';')  
    dataframe = dataframe.rename(columns={'Geographic Area': 'city'})  
    dataframe['city'].replace([' city', ' City', ' CDP', ' town', ' Town', ' village', ' Village'],  
'', regex=True, inplace=True)  
    dataframe['population'].replace(',', '.', regex=True, inplace=True)  
    dataframe['population'].replace('-', np.NaN, regex=True, inplace=True)  
    dataframe.dropna()  
    dataframe['population'] = np.round(dataframe['population'].to_numpy(dtype='float32')).astype(int)  
  
    dataframe.to_csv(index=False, path_or_buf='C:\\Users\\rinos\\OneDrive\\Desktop\\ITPS\\TERZO  
ANNO\\SECONDO SEMESTRE\\Data mining\\progetto\\police shootings\\states\\final states\\' +  
state_name + '.csv')
```

#### 4.2.2.2 Merge dei dataset americani

Al fine di facilitare il *merge* con gli altri dataset, si è provveduto a unire i dataset degli stati americani appena rielaborati in uno solo, in modo da ottenere un elenco di città, accompagnato dal numero dei suoi abitanti.

Per fare questo, è stato implementato il seguente codice che semplicemente aziona un *append* al file finale *all\_states.csv*, per ogni dataset:

```
for state in states_name:
    with open('C:\\Users\\rinos\\OneDrive\\Desktop\\ITPS\\TERZO
ANNO\\SECONDO SEMESTRE\\Data mining\\progetto\\police
shootings\\states\\final states\\' + state + '.csv', 'r') as f:
        temp = f.read()

    with open('C:\\Users\\rinos\\OneDrive\\Desktop\\ITPS\\TERZO
ANNO\\SECONDO SEMESTRE\\Data mining\\progetto\\police
shootings\\states\\final states\\all_states.csv', 'a') as final_file:
        final_file.write('\r')
        final_file.write(temp)
```

### 4.3 Prima integrazione

Terminata l'elaborazione dei primi due dataset ([PoliceKillingsInUS](#) & [dataset americani](#)), si procede ad effettuare il primo *merge*.

Si utilizza quindi il metodo *merge()* della libreria *pandas*, il quale funziona come se si stesse usando il comando *JOIN* in *SQL*, in quanto occorre specificare nei parametri la colonna grazie alla quale è possibile effettuare il collegamento tra i due dataset.

Si è scelto quindi di effettuare il *merge* sull'attributo *city*, e di eliminare i record nulli eventualmente generati dopo l'operazione:

```
merge1 = pd.merge(police_killings, all_cities, on='city', how='left')
merge1.dropna(inplace=True)
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

```
merge1.head()
```

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera	population
Yes	53	M	No	Shelton, WA	Yes	Yes	No	No	9945.0
No	23	M	No	Wichita, KS	No	Yes	No	No	390069.0
Yes	32	M	No	San Francisco, CA	Yes	Yes	No	No	870854.0
Yes	39	M	No	Evans, CO	No	Yes	No	No	20162.0
Yes	18	M	No	Guthrie, OK	No	Yes	No	No	11235.0

Come risultato, il nuovo dataset avrà 2069 record:

```
merge1 length:
2069
```

con alcun valore nullo:

```
merge1.isnull().sum()

armed          0
age            0
gender         0
is_black       0
city           0
signs_of_mental_illness  0
threat         0
flee           0
body_camera    0
population     0
```

## 4.4 dataset PercentOver25CompletedHighSchool

Come visto nella fase di *Data Understanding* ([qui](#)), non ci valori nulli all'interno di questo dataset, tuttavia sappiamo che l'attributo *percent\_completed\_hs* in alcuni casi (197) assume un valore non conforme al tipo dell'attributo, ovvero "-". Quindi, si decide di sostituire ad ogni occorrenza di tale valore, il valore dato dalla media di tutti i data objects. Infine, per uniformare il dataset al *merge1* appena fatto e anche per agevolare la procedura di integrazione, si decide di creare l'attributo *city*, che corrisponde alla formattazione fatta fino ad ora per gli attributi *City & Geographic Area*.

### 4.4.1 Caricamento in memoria del dataset

Come prima cosa, si carica in memoria il dataset tramite il seguente comando:

```
percent_completed_hs = pd.read_csv(r'C:\Users\rinos\OneDrive\Desktop\ITPS\TERZO ANNO\SECONDO SEMESTRE\Data mining\progetto\police shootings\PercentOver25CompletedHighSchool.csv')
```

### 4.4.2 Sostituzione valori

Si procede alla sostituzione dei valori "-" con il valore *NaN*, e ne si mostra il risultato sui primi valori:

```
percent_completed_hs['percent_completed_hs'].replace('-', np.NaN, regex=True, inplace=True)
percent_completed_hs['percent_completed_hs'].value_counts(dropna=False).to_frame().style
```

percent_completed_hs	
100	1301
nan	197
91.7	170
92.9	169
92.5	168
91.3	163
89.3	162

Prima di rimpiazzare i valori *NaN*, si procede con la conversione di tipo, ovvero da *object* a *float*, e successivamente si sostituiscono i valori della media:

```
percent_completed_hs['percent_completed_hs'] = percent_completed_hs['percent_completed_hs'].astype(float)
mean = percent_completed_hs['percent_completed_hs'].mean()
percent_completed_hs['percent_completed_hs'].fillna(value=mean, inplace=True)

percent_completed_hs['percent_completed_hs'] = percent_completed_hs['percent_completed_hs'].round(1)|
```

Si stampano i risultati:

```
percent_completed_hs['percent_completed_hs'].value_counts(dropna=False).to_frame().style
```

percent_completed_hs	
100.0	1301
85.7	325
91.7	170
92.9	169
92.5	168
91.3	163
89.3	162

### 4.4.3 Formattazione nuovo attributo

Come anticipato, si crea un attributo *city*, i cui valori rispetteranno il formato già precedentemente esposto “*nome\_città, codice\_stato*”. Prima di far questo, si rimuove dai valori dell’attributo *city* tutte le parole, o sottostringhe che contengono “city, City, CDP, town, Town, village, Village”.

```
percent_completed_hs['City'].replace([' city', ' City', ' CDP', ' town', ' Town', ' village', ' Village'], '', regex=True, inplace=True)
```

Successivamente, si formattano i valori dell’attributo, in modo da rispettare la forma indicata “*nome\_città, codice\_stato*”:

```
percent_completed_hs['city'] = percent_completed_hs['City'] + ', ' + percent_completed_hs['Geographic Area']
```

Come ultimo passaggio, si eliminano quindi gli attributi *City & state*, in quanto inglobati all’interno del nuovo attributo *city*.

```
percent_completed_hs.drop(['Geographic Area', 'City'], axis=1, inplace=True)
```

Tramite il metodo *head()*, si recuperano i primi valori del dataset, con le rispettive colonne, per verificare l’esecuzione dei comandi:

percent_completed_hs	city
21.2	Abanda, AL
69.1	Abbeville, AL
78.9	Adamsville, AL
81.4	Addison, AL
68.6	Akron, AL

## 4.5 Seconda integrazione

Terminata l'elaborazione di [questo dataset](#), si procede ad effettuare il secondo *merge*.

Si utilizza quindi il metodo *merge()* della libreria *pandas*, come effettuato precedentemente.

Si è scelto quindi di effettuare il *merge* sull'attributo *city*:

```
merge2 = pd.merge(merge1, percent_completed_hs, on='city', how='inner')
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera	population	percent_completed_hs
Yes	53	M	No	Shelton, WA	Yes	Yes	No	No	9945	80.1
No	23	M	No	Wichita, KS	No	Yes	No	No	390069	87.5
Yes	26	M	No	Wichita, KS	No	Yes	Yes	No	390069	87.5
Yes	18	M	No	Wichita, KS	No	Yes	Yes	Yes	390069	87.5
Yes	25	M	No	Wichita, KS	No	Yes	No	No	390069	87.5

Come risultato, il nuovo dataset avrà 2066 record:

```
merge2 length:  
2066
```

con alcun valore nullo:

```
merge2.isnull().sum()  
  
armed      0  
age        0  
gender     0  
is_black   0  
city       0  
signs_of_mental_illness  0  
threat     0  
flee       0  
body_camera  0  
population  0  
percent_completed_hs  0
```

## 4.6 dataset PercentagePeopleBelowPovertyLevel

Questo dataset è simile al precedente, pertanto verranno effettuate le medesime operazioni di elaborazione.

### 4.6.1 Caricamento in memoria del dataset

Come prima cosa, si carica in memoria il dataset tramite il seguente comando:

```
people_below_poverty_level = pd.read_csv(r'C:\Users\rinos\OneDrive\Desktop\ITPS\TERZO ANNO\SECONDO SEMESTRE\Data mining\progetto\police shootings\PercentagePeopleBelowPovertyLevel.csv')
```

### 4.6.2 Sostituzione valori

Si procede alla sostituzione dei valori “-” con il valore *NaN*, e ne si mostra il risultato sui primi valori:

```
people_below_poverty_level['poverty_rate'].replace('-', np.NaN, regex=True, inplace=True)
people_below_poverty_level['poverty_rate'].value_counts(dropna=False).to_frame().style
```

poverty_rate	
0	1464
nan	201
6.7	129
7.4	129
10	128
10.9	128
6.3	126

Prima di rimpiazzare i valori *NaN*, si procede con la conversione di tipo, ovvero da *object* a *float*, e successivamente si sostituiscono i valori della media:

```
people_below_poverty_level['poverty_rate'] = people_below_poverty_level['poverty_rate'].astype(float)
mean = people_below_poverty_level['poverty_rate'].mean()
people_below_poverty_level['poverty_rate'].fillna(value=mean, inplace=True)

people_below_poverty_level['poverty_rate'] = people_below_poverty_level['poverty_rate'].round(1)
```

Si stampano i risultati:

```
people_below_poverty_level['poverty_rate'].value_counts(dropna=False).to_frame().style
```

poverty_rate	
0.0	1464
16.4	276
7.4	129
6.7	129
10.9	128
10.0	128
4.4	126

### 4.6.3 Formattazione nuovo attributo

Come anticipato, si crea un attributo *city*, i cui valori rispetteranno il formato già precedentemente esposto “*nome\_città, codice\_stato*”. Prima di far questo, si rimuove dai valori dell’attributo *city* tutte le parole, o sottostringhe che contengono “city, City, CDP, town, Town, village, Village”.

```
people_below_poverty_level['City'].replace([' city', ' City', ' CDP', ' town', ' Town', ' village', ' Village'], '', regex=True, inplace=True)
```

Successivamente, si formattano i valori dell’attributo, in modo da rispettare la forma indicata “*nome\_città, codice\_stato*”:

```
people_below_poverty_level['city'] = people_below_poverty_level['City'] + ', ' + people_below_poverty_level['Geographic Area']
```

Come ultimo passaggio, si eliminano quindi gli attributi *City & state*, in quanto inglobati all’interno del nuovo attributo *city*.

```
people_below_poverty_level.drop(['Geographic Area', 'City'], axis=1, inplace=True)
```

Tramite il metodo *head()*, si recuperano i primi valori del dataset, con le rispettive colonne, per verificare l’esecuzione dei comandi:

poverty_rate	city
78.8	Abanda, AL
29.1	Abbeville, AL
25.5	Adamsville, AL
30.7	Addison, AL
42.0	Akron, AL



## 4.7 Terza integrazione

Terminata l'elaborazione di [questo dataset](#), si procede ad effettuare il terzo *merge*.

Si utilizza quindi il metodo *merge()* della libreria *pandas*, come effettuato precedentemente.

Si è scelto quindi di effettuare il *merge* sull'attributo *city*:

```
merge3 = pd.merge(merge2, people_below_poverty_level, on='city', how='inner')
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera	population	percent_completed_hs	poverty_rate
Yes	53	M	No	Shelton, WA	Yes	Yes	No	No	9945	80.1	28.6
No	23	M	No	Wichita, KS	No	Yes	No	No	390069	87.5	17.3
Yes	26	M	No	Wichita, KS	No	Yes	Yes	No	390069	87.5	17.3
Yes	18	M	No	Wichita, KS	No	Yes	Yes	Yes	390069	87.5	17.3
Yes	25	M	No	Wichita, KS	No	Yes	No	No	390069	87.5	17.3

Come risultato, il nuovo dataset avrà 2086 record:

```
merge3 length:  
2086
```

con alcun valore nullo:

```
merge3.isnull().sum()  
  
armed          0  
age            0  
gender         0  
is_black       0  
city           0  
signs_of_mental_illness 0  
threat         0  
flee           0  
body_camera    0  
population     0  
percent_completed_hs 0  
poverty_rate   0
```

## 4.8 dataset MedianHouseholdIncome2015

Questo dataset è simile al precedente, pertanto verranno effettuate le medesime operazioni di elaborazione, eccezion fatta per una ulteriore sostituzione da applicare ad alcuni valori dell'attributo *Median Income*, che assumono la forma di "(X)".

### 4.8.1 Caricamento in memoria del dataset

Come prima cosa, si carica in memoria il dataset tramite il seguente comando:

```
people_below_poverty_level = pd.read_csv(r'C:\Users\rinos\OneDrive\Desktop\ITPS\TERZO ANNO\SECONDO SEMESTRE\Data mining\progetto\police shootings\PercentagePeopleBelowPovertyLevel.csv')
```

### 4.8.2 Sostituzione valori

Si procede alla sostituzione dei valori "-" e "(X)" con il valore *NaN*, e ne si mostra il risultato sui primi valori:

```
household_income['Median Income'].replace(['(X)', '-'], np.NaN, regex=True, inplace=True)
household_income['Median Income'].value_counts(dropna=False).to_frame().style
```

Median Income	
nan	1937
38750	136
41250	125
43750	115
46250	112
33750	103
36250	102

Prima di rimpiazzare i valori *NaN*, si procede con la conversione di tipo, ovvero da *object* a *float*, successivamente si sostituiscono i valori della media e si riconverte il tipo ad *int*:

```
household_income['Median Income'] = household_income['Median Income'].astype(float)
mean = household_income['Median Income'].mean()
household_income['Median Income'].fillna(value=mean, inplace=True)
household_income['Median Income'] = household_income['Median Income'].astype(int)
```

Si stampano i risultati:

```
household_income['Median Income'].value_counts(dropna=False).to_frame().style
```

Median Income	
51177	1937
38750	136
41250	125
43750	115
46250	112
33750	103
36250	102

### 4.8.3 Renaming attributo Median Income

Prima di continuare, si procede al rename dell'attributo *Median Income*, per adattarlo al format dei nomi utilizzato fino ad adesso. Si cambia quindi il nome in "*median\_income*":

```
household_income = household_income.rename(columns={'Median Income': 'median_income'})
```

### 4.8.4 Formattazione nuovo attributo

Come anticipato, si crea un attributo *city*, i cui valori rispetteranno il formato già precedentemente esposto "*nome\_città, codice\_stato*". Prima di far questo, si rimuove dai valori dell'attributo *city* tutte le parole, o sottostringhe che contengono "city, City, CDP, town, Town, village, Village".

```
household_income['City'].replace([' city', ' City', ' CDP', ' town', ' Town', ' village', ' Village'], '', regex=True, inplace=True)
```

Successivamente, si formattano i valori dell'attributo, in modo da rispettare la forma indicata "*nome\_città, codice\_stato*":

```
household_income['city'] = household_income['City'] + ', ' + household_income['Geographic Area']
```

Come ultimo passaggio, si eliminano quindi gli attributi *City & state*, in quanto inglobati all'interno del nuovo attributo *city*.

```
household_income.drop(['Geographic Area', 'City'], axis=1, inplace=True)
```

Tramite il metodo *head()*, si recuperano i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione dei comandi:

median_income	city
11207	Abanda, AL
25615	Abbeville, AL
42575	Adamsville, AL
37083	Addison, AL
21667	Akron, AL

## 4.9 Quarta integrazione

Terminata l'elaborazione di [questo dataset](#), si procede ad effettuare il quarto *merge*.

Si utilizza quindi il metodo *merge()* della libreria *pandas*, come effettuato precedentemente.

Si è scelto quindi di effettuare il *merge* sull'attributo *city*:

```
merge4 = pd.merge(merge3, household_income, on='city', how='inner')
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera	population	percent_completed_hs	poverty_rate	median_income	
Yes	53	M	No	Shelton, WA		Yes	Yes	No	No	9945	80.1	28.6	37072
No	23	M	No	Wichita, KS		No	Yes	No	No	390069	87.5	17.3	45947
Yes	26	M	No	Wichita, KS		No	Yes	Yes	No	390069	87.5	17.3	45947
Yes	18	M	No	Wichita, KS		No	Yes	Yes	Yes	390069	87.5	17.3	45947
Yes	25	M	No	Wichita, KS		No	Yes	No	No	390069	87.5	17.3	45947

Come risultato, il nuovo dataset avrà 2126 record:

```
merge4 length:
2126
```

con alcun valore nullo:

```
merge4.isnull().sum()

armed          0
age            0
gender         0
is_black       0
city           0
signs_of_mental_illness  0
threat         0
flee           0
body_camera    0
population     0
percent_completed_hs  0
poverty_rate   0
median_income  0
```

## 4.10 dataset ShareRaceByCity

*ShareRaceByCity* è l'ultimo dataset da rielaborare. Come già visto, contiene gli attributi ritrovati in precedenza *Geographic area & City*, su cui si effettuerà la medesima rielaborazione effettuata nei paragrafi precedenti per gli altri dataset, e anche una serie di attributi che dovrebbero risultare di tipo *float*, quindi *numerici continui*, ma che invece risultano come *object*.

Ai fini dell'analisi proposta però, questa serie di attributi risulta poco utile allo stato attuale. Dato che precedentemente, per il [dataset PoliceKillingsUS si è deciso di eliminare l'attributo \*race\*](#) (contenente come valori la lettera iniziale per ciascuna etnia d'origine) e di sostituirlo con *is\_black*, in questo dataset si lavorerà alla stessa maniera.

Il risultato finale sarà dunque la sostituzione degli attributi *share\_white*, *share\_black*, *share\_asian*, *share\_hispanic*, *share\_native\_american* con solo due attributi, ovvero *share\_black*, *share\_not\_black*.

Prima di raggiungere tale risultato, verrà effettuata una *cleaning* dell'attributo *share\_black*, il quale possiede i valori anomali individuati nel dataset precedente, ovvero “-” e “(X)”.

### 4.10.1 Caricamento in memoria del dataset

Come prima cosa, si carica in memoria il dataset tramite il seguente comando:

```
share_race_by_city = pd.read_csv(r'C:\Users\rinos\OneDrive\Desktop\ITPS\TERZO ANNO\SECONDO SEMESTRE\Data mining\progetto\police shootings\ShareRaceByCity.csv')
```

### 4.10.2 Sostituzione valori

Si procede alla sostituzione dei valori “-” e “(X)” con il valore *NaN*:

```
share_race_by_city['share_black'].replace(['(X)', '-'], np.NaN, regex=True, inplace=True)
```

Prima di rimpiazzare i valori *NaN*, si procede con la conversione di tipo, ovvero da *object* a *float*:

```
share_race_by_city['share_black'] = share_race_by_city['share_black'].astype(float)
mean = share_race_by_city['share_black'].mean()
share_race_by_city['share_black'].fillna(value=mean, inplace=True)

share_race_by_city['share_black'] = share_race_by_city['share_black'].round(1)
```

Si stampano i risultati:

```
share_race_by_city['share_black'].value_counts().to_frame().style
```

	share_black
0.0	6587
0.2	1346
0.3	1333
0.4	1256
0.5	1078
0.6	1010
0.1	898

### 4.10.3 Creazione nuovo attributo *share\_not\_black*

Come anticipato, per ottenere i valori di questo attributo si è semplicemente pensato di applicare una differenza, ovvero sottrarre a 100.00 (la percentuale massima) il valore che assume *share\_black*, in modo che appunto siano complementari. Per far questo, usiamo il seguente comando:

```
share_race_by_city['share_not_black'] = 100.00 - share_race_by_city['share_black']
```

Dunque, è ora possibile eliminare tutti gli altri attributi, già citati in precedenza:

```
share_race_by_city.drop(['share_white', 'share_native_american', 'share_asian', 'share_hispanic'], axis=1, inplace=True)
```

Si stampano ora i risultati, tramite il metodo *head()*:

Geographic area	City	share_black	share_not_black
AL	Abanda CDP	30.2	69.8
AL	Abbeville city	41.4	58.6
AL	Adamsville city	44.9	55.1
AL	Addison town	0.1	99.9
AL	Akron town	86.5	13.5

### 4.10.4 Formattazione nuovo attributo

Come anticipato, si crea un attributo *city*, i cui valori rispetteranno il formato già precedentemente esposto “*nome\_città, codice\_stato*”. Prima di far questo, si rimuove dai valori dell’attributo *city* tutte le parole, o sottostringhe che contengono “city, City, CDP, town, Town, village, Village”.

```
share_race_by_city['City'].replace([' city', ' City', ' CDP', ' town', ' Town', ' village', ' Village'], '', regex=True, inplace=True)
```

Successivamente, si formattano i valori dell’attributo, in modo da rispettare la forma indicata “*nome\_città, codice\_stato*”:

```
share_race_by_city['city'] = share_race_by_city['City'] + ', ' + share_race_by_city['Geographic area']
```

Come ultimo passaggio, si eliminano quindi gli attributi *City & state*, in quanto inglobati all’interno del nuovo attributo *city*.

```
share_race_by_city.drop(['Geographic area', 'City'], axis=1, inplace=True)
```

Tramite il metodo *head()*, si recuperano i primi valori del dataset, con le rispettive colonne, per verificare l’esecuzione dei comandi:

share_black	share_not_black	city
30.2	69.8	Abanda, AL
41.4	58.6	Abbeville, AL
44.9	55.1	Adamsville, AL
0.1	99.9	Addison, AL
86.5	13.5	Akron, AL

## 4.11 Quinta e ultima integrazione

Terminata l'elaborazione di [questo dataset](#), si procede ad effettuare il quinto *merge*, ovvero l'ultimo, che risulterà essere il dataset finale.

Si utilizza quindi il metodo *merge()* della libreria *pandas*, come effettuato precedentemente.

Si è scelto quindi di effettuare il *merge* sull'attributo *city*:

```
merge4 = pd.merge(merge3, household_income, on='city', how='inner')
```

Tramite il metodo *head()*, recuperiamo i primi valori del dataset, con le rispettive colonne, per verificare l'esecuzione del comando:

armed	age	gender	is_black	city	signs_of_mental_illness	threat	flee	body_camera	population	percent_completed_hs	poverty_rate	median_income	share_black	share_not_black
Yes	53	M	No	Shelton, WA		Yes	Yes	No	9945	80.1	28.6	37072	0.8	99.2
No	23	M	No	Wichita, KS		No	Yes	No	390069	87.5	17.3	45947	11.5	88.5
Yes	26	M	No	Wichita, KS		No	Yes	Yes	390069	87.5	17.3	45947	11.5	88.5
Yes	18	M	No	Wichita, KS		No	Yes	Yes	390069	87.5	17.3	45947	11.5	88.5
Yes	25	M	No	Wichita, KS		No	Yes	No	390069	87.5	17.3	45947	11.5	88.5

Come risultato, il nuovo dataset avrà 2204 record:

```
dataset length:  
2204
```

con alcun valore nullo:

```
dataset.isnull().sum()  
  
armed          0  
age            0  
gender         0  
is_black       0  
city           0  
signs_of_mental_illness  0  
threat         0  
flee           0  
body_camera    0  
population     0  
percent_completed_hs  0  
poverty_rate   0  
median_income  0  
share_black    0  
share_not_black 0
```

## 5. Modeling

### 5.1 Prime considerazioni

Una volta costruito il dataset, si può importarlo su *Weka*, il software scelto per questa fase del processo di KD (scaricabile da qui <sup>[9]</sup>).

Prima di iniziare con la generazione del modello, è stato ritenuto opportuno ritoccare nuovamente il dataset con delle operazioni comuni ad entrambi i task di data mining descritti.

Alla luce del lavoro fatto, ne vien fuori che, come anche previsto in una prima fase del processo di KD ([qui](#)), il **numero delle vittime di etnia afroamericana** presente nel dataset è **decisamente inferiore** a quello delle altre etnie, e questo è una diretta riflessione della eterogeneità della popolazione statunitense. Si è pensato dunque che l'eventuale (gli eventuali) modello (modelli) di classificazione e descrizione potesse (potessero) in qualche modo risentire di questa disparità, e che potesse (potessero) dunque offrire dei risultati erranei o quanto meno inverosimili o poco aderenti alla realtà dei fatti. Pertanto, si è deciso di rivalutare il numero di istanze totali del dataset, cercando di diminuire quanto più possibile questo *gap*. Tale operazione è stata svolta in *Weka*, tramite l'applicazione del **Filtro SpreadSubsample**.

Di seguito, le operazioni preliminari.

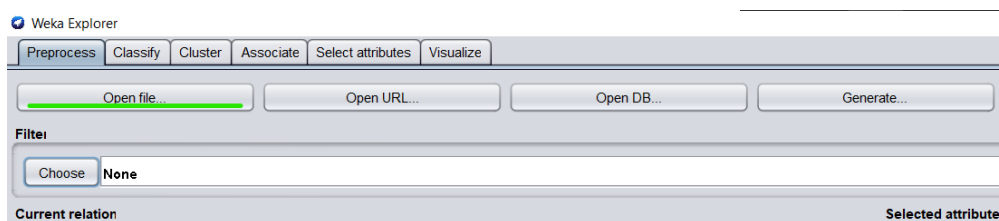
#### 5.1.1 Caricamento del dataset

Si inizia con l'import del dataset in memoria.

Per farlo, si apre il *Weka Explorer* presente nella finestra di *Weka GUI Chooser*:



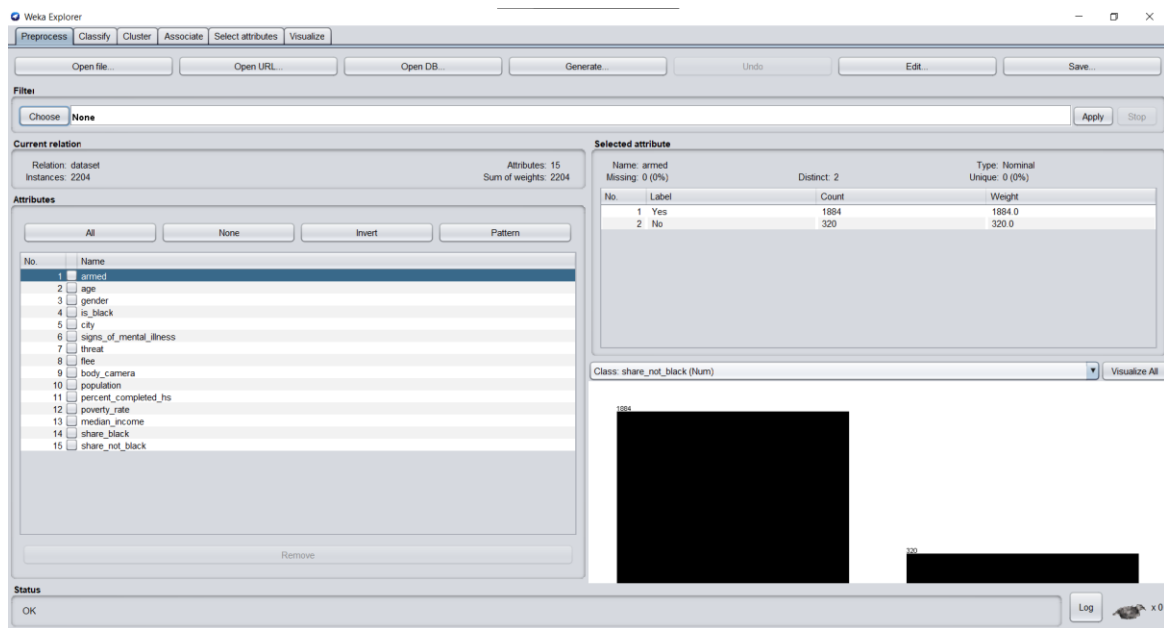
e nella sezione in alto, si seleziona *Open file*, e si naviga nella directory dove è presente il dataset:



<sup>9</sup> [https://waikato.github.io/weka-wiki/downloading\\_weka](https://waikato.github.io/weka-wiki/downloading_weka)



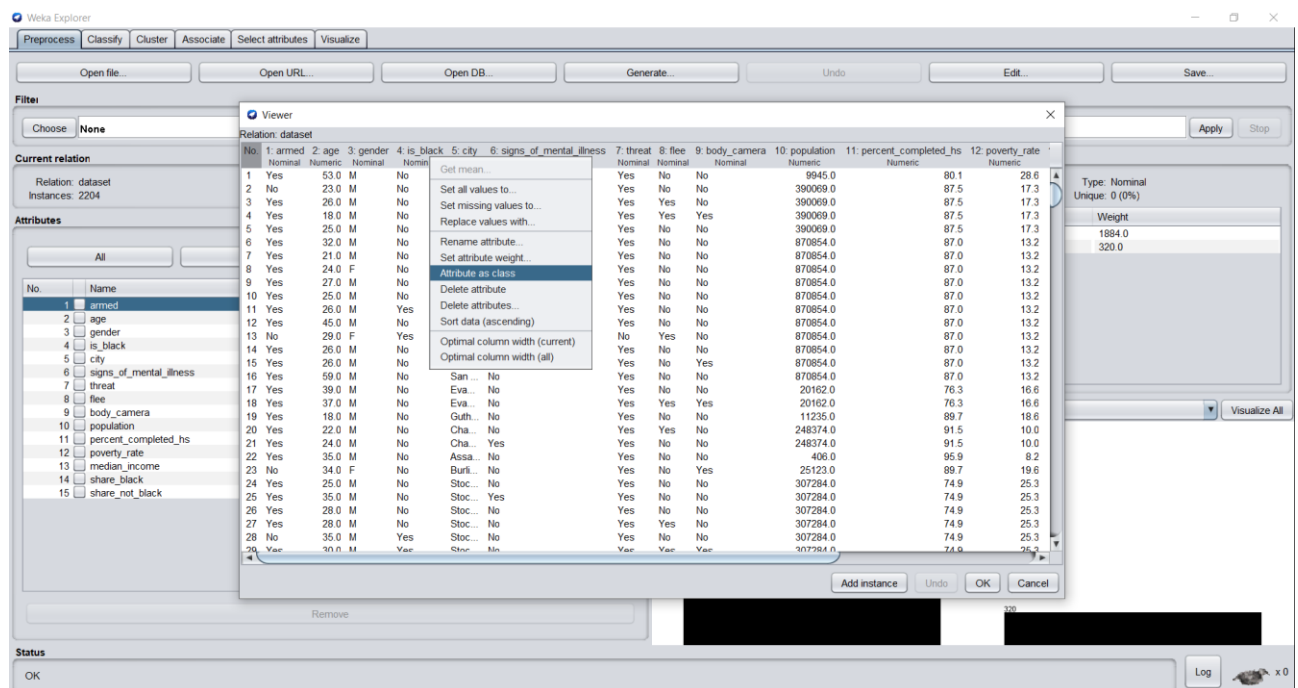
La finestra principale si mostra così:



### 5.1.2 Selezione attributo di classe

In automatico, *Weka* seleziona come attributo di classe l'ultimo della lista, quello indicato dall'indice più grande numericamente.

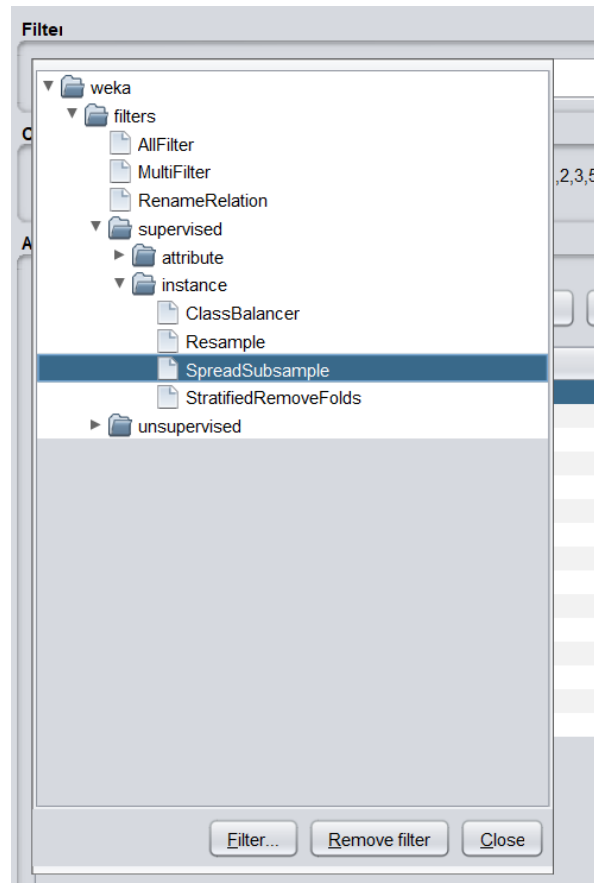
Settiamo quindi come attributo di classe ***is\_black***. Per farlo, si clicca in alto a destra su *Edit*, e nella finestra che appare, al cui interno è presente il dataset in forma tabellare, cliccare con il tasto destro sull'attributo da modificare, dopodiché lo si setta come etichetta di classe:



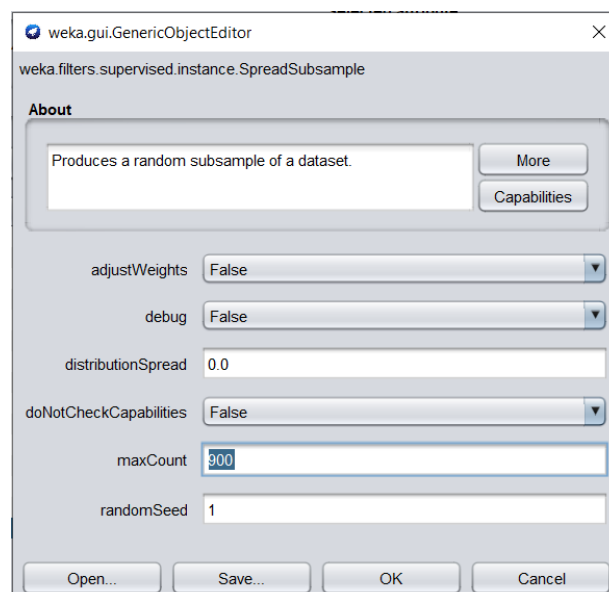
### 5.1.3 Applicazione filtro SpreadSubsample

Adesso è possibile effettuare quella procedura atta a ridurre il gap tra il numero di istanze dei valori *Yes* & *No* dell'attributo classe *is\_black*.

Per farlo, si clicca sul bottone *Choose* della sezione *Filter* in alto, e nella directory *supervised* > *instance* si sceglie *SpreadSubsample*:



Nella finestra delle impostazioni del filtro, si sceglie come valore da attribuire a *maxCount* (ovvero il numero massimo di istanze per ogni valore di classe), **900** (considerando che, il valore *Yes* = 1667 & *No* = 537). In questo modo, si è quantomeno ridotto il gap.



## 5.2 Prediction

Obiettivo del *predictive data mining task* è, come già illustrato [qui](#), quello di costruire un modello di classificazione che sia in grado di stimare, dati una serie di parametri ben definiti, se la vittima di un intervento fatale da parte della polizia statunitense sia di etnia afroamericana o meno.

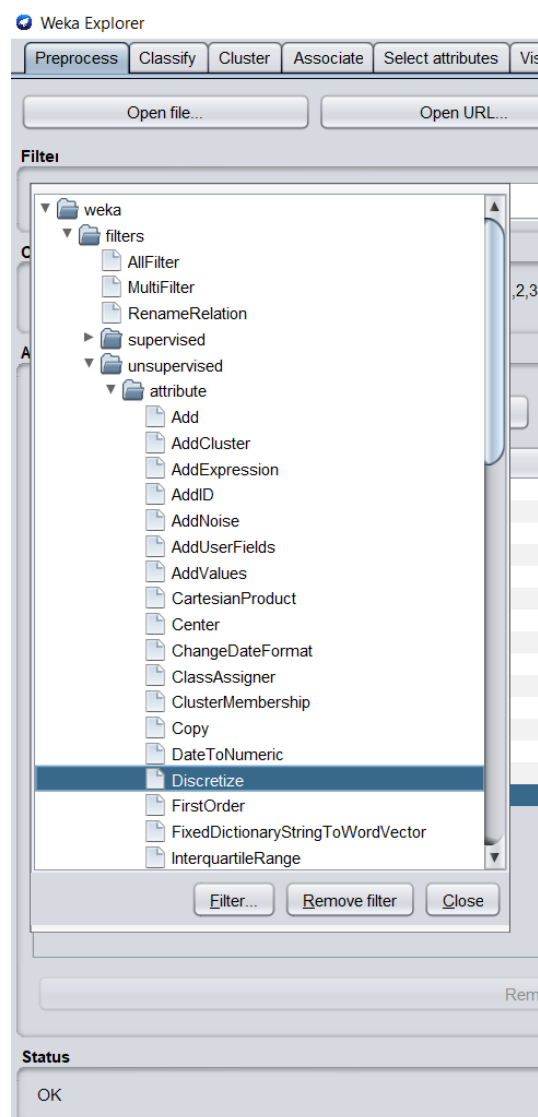
Si decide quindi di provare a costruire più modelli, utilizzando classificatori diversi, andando poi a definire quello più conforme all'obiettivo prefissato nella fase di **Evaluation**.

Prima di iniziare, in virtù del fatto che diversi attributi assumono range di valori decisamente ampi, si è pensato di applicare, per il task di predizione, il filtro **Discretization**, il quale appunto va a suddividere i valori numerici presenti all'interno di una colonna in un determinato numero di *bins*. Gli attributi presi in considerazione sono *age*, *population*, *percent\_completed\_hs*, *poverty\_rate*, e *median\_income*, *share\_black* e *share\_not\_black*.

### 5.2.1 Elaborazioni finali

#### 5.2.1.1 Discretization

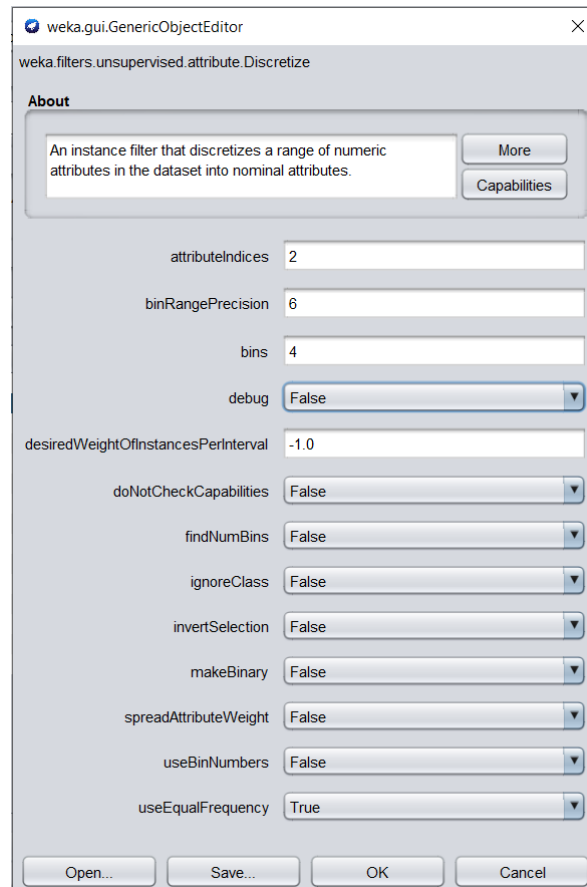
Per applicare il filtro di discretizzazione, si clicca sul bottone *Choose* della sezione *Filter* in alto, e nella directory *unsupervised > attribute* si sceglie *Discretize*:



Nei successivi paragrafi, verrà mostrata la finestra delle opzioni utilizzate, attributo per attributo.

Si è deciso, come detto, di discretizzare l'attributo **age**, tramite **equal frequency**, in **4 bins**, in modo tale da avere degli intervalli ben definiti e ricchi di data objects alla stessa maniera, visto che il range di valori è abbastanza ampio e, con **equal width**, si rischierebbe di avere **bins** con un numero molto ridotto di record.

Nella finestra delle impostazioni del filtro, si definisce l'indice dell'attributo da discretizzare (2), il numero di bins in cui dividere i valori (4), e infine si spunta il flag **useEqualFrequency**:



Si è deciso, come detto, di discretizzare gli attributi **population**, **percent\_completed\_hs**, **poverty\_rate**, **median\_income**, **share\_black** e **share\_not\_black** tramite **equal frequency**, in **5 bins**, in modo tale da avere degli intervalli ben definiti e ricchi di data objects alla stessa maniera, visto che il range di valori è abbastanza ampio e, con **equal width**, si rischierebbe di avere **bins** con un numero molto ridotto di record.

Nella finestra delle impostazioni del filtro, si definisce l'intervallo degli indici degli attributi da discretizzare (9-12), il numero di bins in cui dividere i valori (5), e infine si spunta il flag *useEqualFrequency*:

The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.filters.unsupervised.attribute.Discretize' filter. The 'About' section contains a description: 'An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.' Below this, various settings are displayed as text boxes or dropdown menus. The 'attributeIndices' is set to '9-14', 'binRangePrecision' to '6', and 'bins' to '5'. The 'debug' dropdown is set to 'False'. The 'desiredWeightOfInstancesPerInterval' is set to '-1.0'. The 'doNotCheckCapabilities', 'findNumBins', 'ignoreClass', 'invertSelection', 'makeBinary', 'spreadAttributeWeight', 'useBinNumbers', and 'useEqualFrequency' dropdowns are all set to 'False', except for 'useEqualFrequency' which is set to 'True'. At the bottom, there are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'.

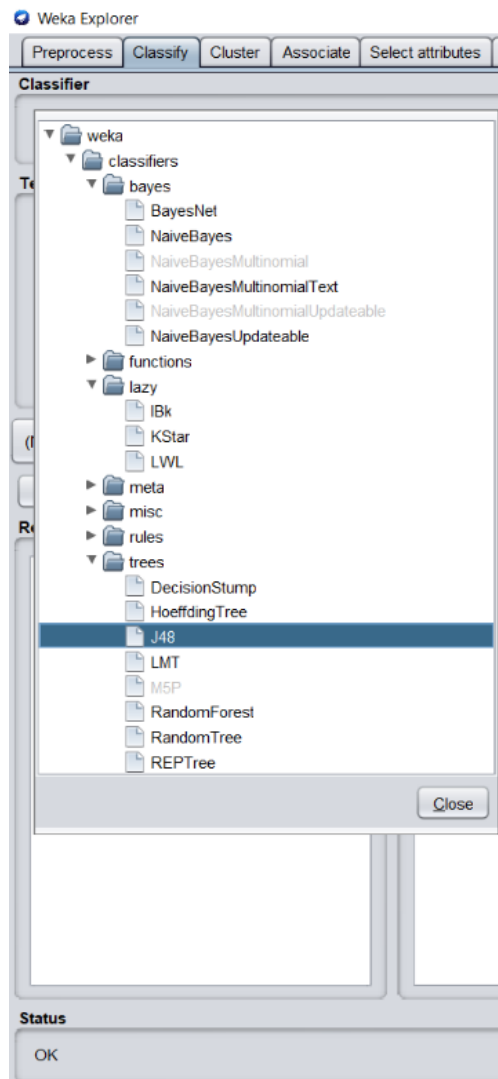
Property	Value
attributeIndices	9-14
binRangePrecision	6
bins	5
debug	False
desiredWeightOfInstancesPerInterval	-1.0
doNotCheckCapabilities	False
findNumBins	False
ignoreClass	False
invertSelection	False
makeBinary	False
spreadAttributeWeight	False
useBinNumbers	False
useEqualFrequency	True

### 5.2.1.1 Divisione del dataset e avvio procedura

Inoltre, non avendo a disposizione un test set, si è deciso di splittare il dataset in due parti in misura percentuale:

- 75% per il *training set*
- 25% per il *test set*.

Per provare a costruire un modello, ci si è recati al tab *Classify* in alto, e, cliccando sul bottone *Choose*, è possibile navigare nelle diverse directory per scegliere il classificatore opportuno:

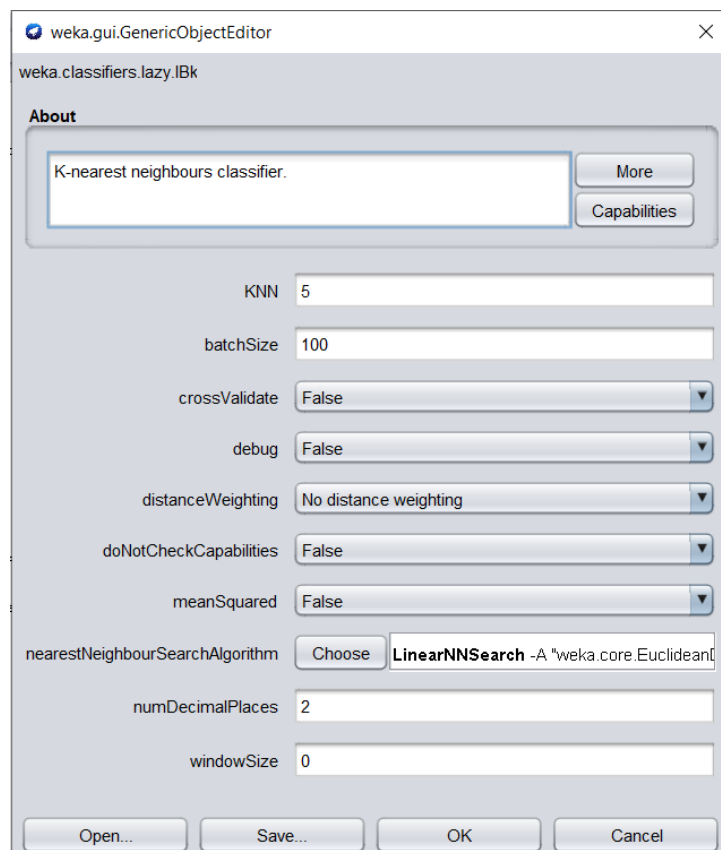


### 5.2.2 K-NN

Il primo classificatore usato è il *K-NN* → K-Nearest-Neighbor.

Definito come *lazy learner*, in *Weka* identificato dal nome **IBk**, tale classificatore rientra nella categoria degli *instance based classifiers*, ovvero quei classificatori che non generano un modello a partire dal training set, ma usano direttamente i suoi attributi per poter stimare i valori o del testing set oppure quelli mancanti di classe/target. Di fatto, esso utilizza le istanze più vicine a quella in esame per assegnare l'etichetta di classe.

Elemento fondamentale di questo classificatore è dunque la scelta del *K*, per cui, dopo diversi tentativi, si è constatato che il valore ideale risulta **pari a 5**. Si riportano di seguito i parametri impostati (lasciato tutto a default tranne il numero di *K*):



The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.classifiers.lazy.IBk' classifier. The 'About' section contains the text 'K-nearest neighbours classifier.' with 'More' and 'Capabilities' buttons. The main configuration area includes the following parameters:

- KNN: 5
- batchSize: 100
- crossValidate: False
- debug: False
- distanceWeighting: No distance weighting
- doNotCheckCapabilities: False
- meanSquared: False
- nearestNeighbourSearchAlgorithm: Choose LinearNNSearch -A "weka.core.Euclidean"
- numDecimalPlaces: 2
- windowSize: 0

At the bottom are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'.

Di seguito i risultati ottenuti:

```
IB1 instance-based classifier
using 5 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.05 seconds

=== Summary ===

Correctly Classified Instances      276          76.8802 %
Incorrectly Classified Instances    83           23.1198 %
Kappa statistic                    0.5027
Mean absolute error                 0.3008
Root mean squared error             0.4025
Relative absolute error             64.0025 %
Root relative squared error         82.7094 %
Total Number of Instances          359

=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,842	0,348	0,795	0,842	0,818	0,504	0,832	0,872	No
	0,652	0,158	0,720	0,652	0,684	0,504	0,832	0,731	Yes
Weighted Avg.	0,769	0,275	0,766	0,769	0,766	0,504	0,832	0,818	

```
=== Confusion Matrix ===

  a   b  <-- classified as
186  35 |  a = No
 48  90 |  b = Yes
```

Sebbene tale classificatore solitamente sia poco preciso nella valutazione delle istanze, in questo caso, come si evince dal *summary*, ne risulta una *accuracy* del 76%. Inoltre, studiando i valori di *Precision* e *Recall*, notiamo come essi denotino una buona performance del modello: *Precision* e *Recall*, che solitamente giocano un *tug of war*, risultano ben bilanciati, pari a 0,766 e 0,769.

Si ricorda che la precision corrisponde a  $\frac{TP}{TP+FP}$ , ovvero va a identificare a quanto ammonta la proporzione di identificazioni positive effettivamente corrette; la *recall* corrisponde a  $\frac{TP}{TP+FN}$ , ovvero va a identificare a quanto ammonta la proporzione di identificazioni effettivamente positive che sono state identificate correttamente.

Buono anche il valore della *F-Measure*, che si attesta a 0,766. Si ricorda che la *F-Measure* è una misura che indica la media armonica di *Precision* e *Recall*, ed è una misura utile per giudicare la bontà di un modello di classificazione. Pertanto più il suo valore si avvicina ad 1, migliore risulterà il modello.



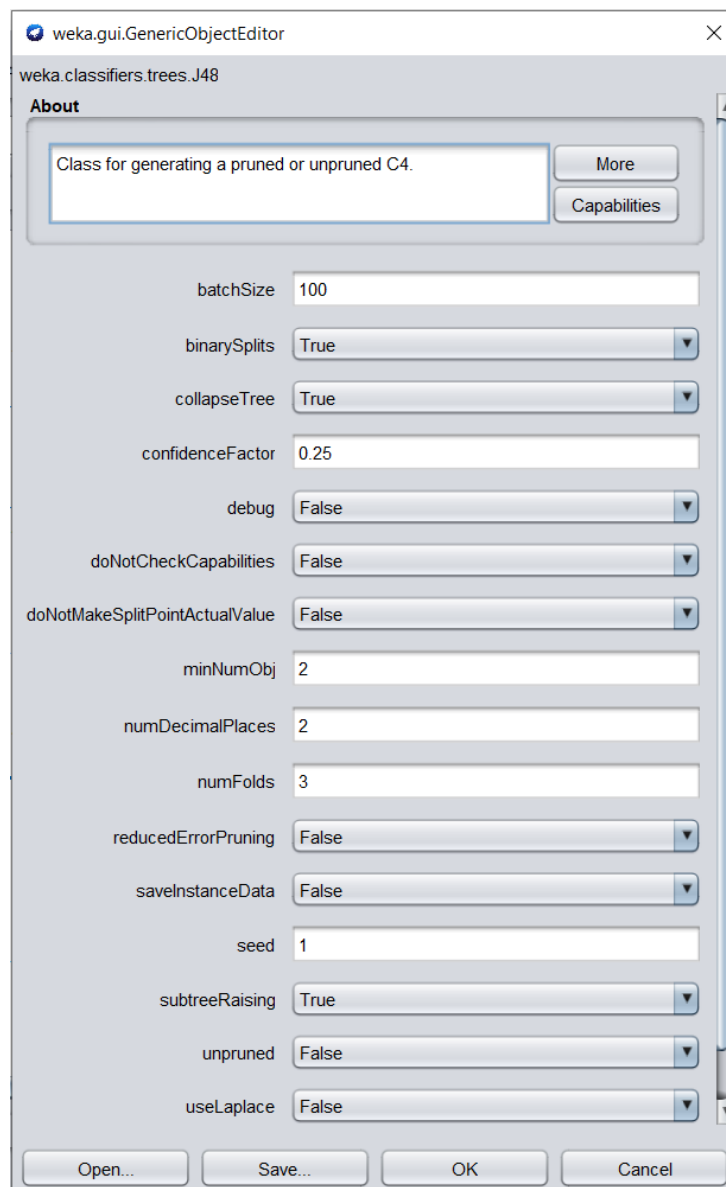
### 5.2.3 C4.5 Classification Tree – J48

Il secondo classificatore usato è il *classification tree*. Definito come *eager learner*, in *Weka* identificato dal nome **J48**, tale classificatore utilizza un albero di decisione per riuscire a classificare un dato elemento, date in input una serie di variabili.

Essenzialmente, in ogni nodo dell'albero viene valutata una condizione, solitamente corrisponde alla valutazione di un attributo: ogni condizione può splittarsi in altri nodi, oppure può splittarsi in foglie, ovvero in parti terminali dell'albero che non hanno più ramificazioni. Le foglie corrispondono alle etichette di classe.

È uno dei classificatori più usati, in quanto è considerato tra quelli più intellegibili e semplici.

Di seguito le impostazioni usate (tutto a default tranne la spunta su *binarySplits* impostata su *True*):



Di seguito i risultati ottenuti:

```
Number of Leaves :      41

Size of the tree :      81

Time taken to build model: 0.03 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances      282          78.5515 %
Incorrectly Classified Instances    77          21.4485 %
Kappa statistic                    0.5437
Mean absolute error                 0.3121
Root mean squared error             0.411
Relative absolute error             66.3946 %
Root relative squared error        84.448 %
Total Number of Instances          359

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,837    0,297    0,819     0,837    0,828      0,544    0,786    0,805    No
                0,703    0,163    0,729     0,703    0,716      0,544    0,786    0,676    Yes
Weighted Avg.   0,786    0,246    0,784     0,786    0,785      0,544    0,786    0,755

=== Confusion Matrix ===

  a    b  <-- classified as
185  36 |    a = No
 41   97 |    b = Yes
```

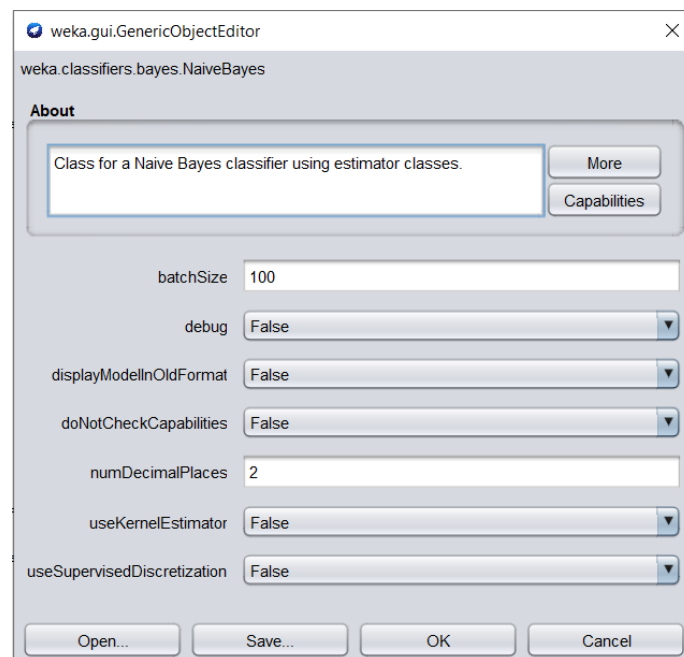
Come si evince dal *summary*, il classificatore ha un *accuracy* del 78%. Anche i valori di *Precision* e *Recall* denotano un buon risultato, e si attestano rispettivamente a 0,784 e 0,786, mentre la F-Measure si attesta a 0,785.

### 5.2.4 Naive Bayes

Tale classificatore è diretto discendente del *teorema di Bayes*, il quale viene appunto utilizzato per la creazione del modello.

Il classificatore si articola in tre fasi differenti: *calcolo della probabilità della classe*, in cui viene effettuato il conteggio delle istanze che appartengono a ciascuna classe, e ognuno di questi conteggi viene diviso per il numero totale di istanze; *calcolo della probabilità condizionata*, in cui si applica il *teorema di Bayes*; *presa di decisione*, in cui si calcola la probabilità della classe di appartenenza di un'istanza sconosciuta o presente nel test set.

Di seguito le impostazioni usate (tutto a default):



Di seguito i risultati ottenuti:

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances	277	77.1588 %
Incorrectly Classified Instances	82	22.8412 %
Kappa statistic	0.5187	
Mean absolute error	0.252	
Root mean squared error	0.4219	
Relative absolute error	53.6228 %	
Root relative squared error	86.704 %	
Total Number of Instances	359	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,810	0,290	0,817	0,810	0,814	0,519	0,847	0,902	No
	0,710	0,190	0,700	0,710	0,705	0,519	0,847	0,763	Yes
Weighted Avg.	0,772	0,251	0,772	0,772	0,772	0,519	0,847	0,849	

=== Confusion Matrix ===

```
a  b  <-- classified as
179 42 |  a = No
40  98 |  b = Yes
```

Come si evince dal *summary*, il classificatore ha un *accuracy* del 77%. Anche i valori di *Precision* e *Recall* denotano un buon risultato, e si attestano entrambi a 0,772, idem per la *F-Measure*.

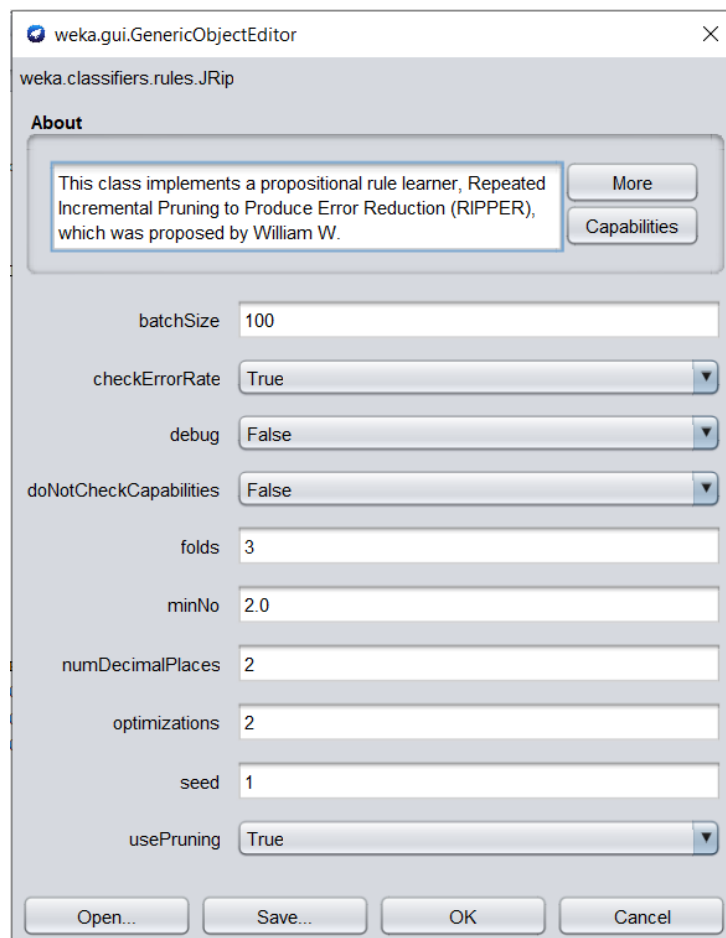
### 5.2.5 Ripper – JRipper

L'ultimo classificatore utilizzato è il *Ripper*, su *Weka* identificato come *JRipper*. Tale classificatore è diverso dai precedenti perché utilizza un set di regole per generare un modello di classificazione. In particolare, una regola si presenta nella forma *if...then*, in cui la parte *if* è definita *antecedente* e contiene una o più condizioni legate da *and logici*, mentre la parte *then* è definita *conseguente* e contiene l'etichetta di classe.

Ogni condizione valuta dunque un attributo del *training set*: l'algoritmo quindi ha come obiettivo quello di costruire delle regole che abbiano delle condizioni tali da coprire il più possibile dei data objects.

L'algoritmo *Ripper* utilizzato parte da un set di regole vuoto e, di volta in volta, aggiunge le regole che più massimizzano il guadagno di informazione.

Di seguito le impostazioni usate (tutto a default):



Di seguito i risultati ottenuti:

```
JRIP rules:
=====

(share_black = '(30.75-inf)') => is_black=Yes (287.0/70.0)
(share_black = '(15.05-30.75]') and (population = '(486266.5-inf)') => is_black=Yes (80.0/26.0)
(share_black = '(15.05-30.75]') and (age = '(-inf-25.5]') => is_black=Yes (54.0/19.0)
(share_black = '(15.05-30.75]') and (median_income = '(55803.5-inf)') and (armed = No) => is_black=Yes (10.0/0.0)
(share_black = '(15.05-30.75]') and (age = '(25.5-33.5]') => is_black=Yes (50.0/22.0)
=> is_black=No (956.0/193.0)

Number of Rules : 6
```

```
Time taken to build model: 0.06 seconds
```

```
=== Evaluation on test split ===
```

```
Time taken to test model on test split: 0 seconds
```

```
=== Summary ===
```

Correctly Classified Instances	280	77.9944 %
Incorrectly Classified Instances	79	22.0056 %
Kappa statistic	0.5306	
Mean absolute error	0.3429	
Root mean squared error	0.4105	
Relative absolute error	72.9539 %	
Root relative squared error	84.3532 %	
Total Number of Instances	359	

```
=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,837	0,312	0,811	0,837	0,824	0,531	0,773	0,789	No
	0,688	0,163	0,725	0,688	0,706	0,531	0,773	0,645	Yes
Weighted Avg.	0,780	0,254	0,778	0,780	0,779	0,531	0,773	0,734	

```
=== Confusion Matrix ===
```

```
  a   b  <-- classified as
185  36 |   a = No
 43  95 |   b = Yes
```

Come si evince dal *summary*, il classificatore ha un *accuracy* del 77%, quasi 78%. Anche i valori di *Precision* e *Recall* denotano un buon risultato, e si attestano rispettivamente a 0,778 e 0,780, mentre la *F-Measure* si attesta a 0,779.

## 5.3 Description

Obiettivo del *descriptive data mining task* è, come illustrato [qui](#), quello di provare ad individuare dei cluster, o degli schemi concreti, che emergano dai dati, per verificare la presenza di un **racial bias** negli interventi fatali compiuti dalla polizia statunitense.

Si decide quindi di utilizzare la tecnica di clustering ***K-means***, il quale ha come obiettivo quello di individuare *K cluster* mutuamente esclusivi tra loro.

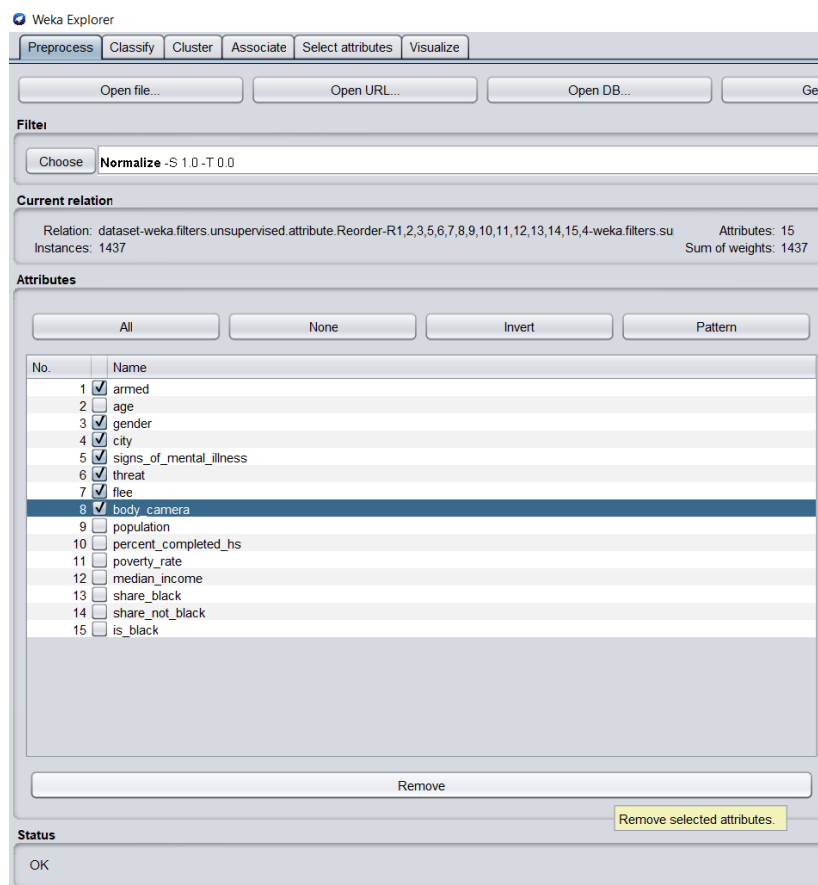
Prima di iniziare, in virtù del fatto che il *K-means* funziona meglio con attributi numerici, visto che per calcolare le distanze tra i data objects utilizza la *Euclidean distance*, si è deciso, per questo task di descrizione, di rimuovere tutti gli attributi *categorici* presenti nel dataset, e di lasciare invariati quelli *numerici*, al fine anche di ottenere un basso valore per il *SSE*.

Si è valutata anche la possibilità di utilizzare la tecnica del *OneHot Encoding* per questi attributi. Tuttavia, dopo alcune analisi, si è scartata tale opzione per via del fatto che avrebbe incrementato di molto la *dimensionalità* del dataset, (si pensi all'attributo *city*) e che quindi avrebbe compromesso la bontà del risultato del *K-means*, facendo impennare il valore del *SSE*.

Gli attributi presi in considerazione sono dunque *armed*, *gender*, *city*, *signs\_of\_mental\_illness*, *threat*, *flee*, *body\_camera*.

### 5.3.1 Elaborazioni finali con rimozione di attributi

Per la rimozione degli attributi, è sufficiente spuntare le righe a loro corrispondenti nella schermata principale di *Weka Explorer*, e cliccare sul bottone *Remove* posto in basso:



### 5.3.2 K-means – SimpleKMeans

Come già accennato [nel paragrafo introduttivo](#) di questa sezione, si è deciso di utilizzare il *K-means*, in *Weka* denominato *SimpleKMeans*, per trovare dei gruppi di data objects distinti.

Questa tecnica necessita di sapere, a priori, il numero esatto di cluster in cui suddividere i data objects, in modo tale da calcolare accuratamente i *centroidi*. Come anticipato, l'algoritmo utilizzato è di tipo iterativo, e procede in questo modo:

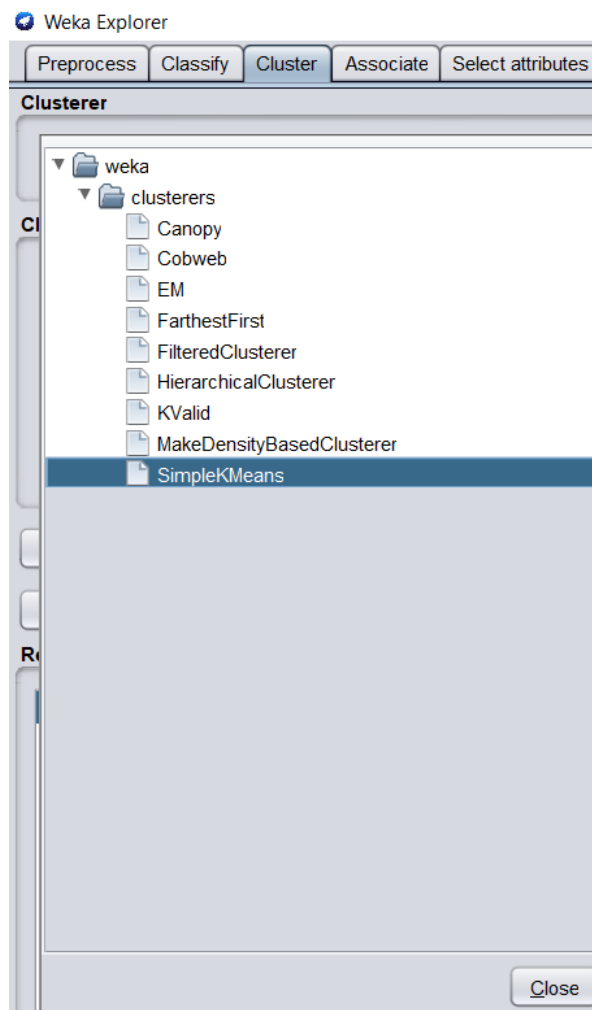
- Vengono scelti randomicamente  $k$  centroidi tra gli esempi.
- Per ogni data object, viene calcolata la sua distanza con i centroidi.
- Il data object viene assegnato al cluster con la distanza minima.
- Terminati i data object, vengono ricalcolati i centroidi e il processo ricomincia.
- Si continua fino a che i centroidi non cambiano più.

Per la scelta del numero di cluster, si è pensato di assegnare  $K = 2$ , per via del fatto che l'attributo di classe *is\_black* assume appunto due valori.

Inoltre si è deciso di splittare il dataset in due parti in misura percentuale:

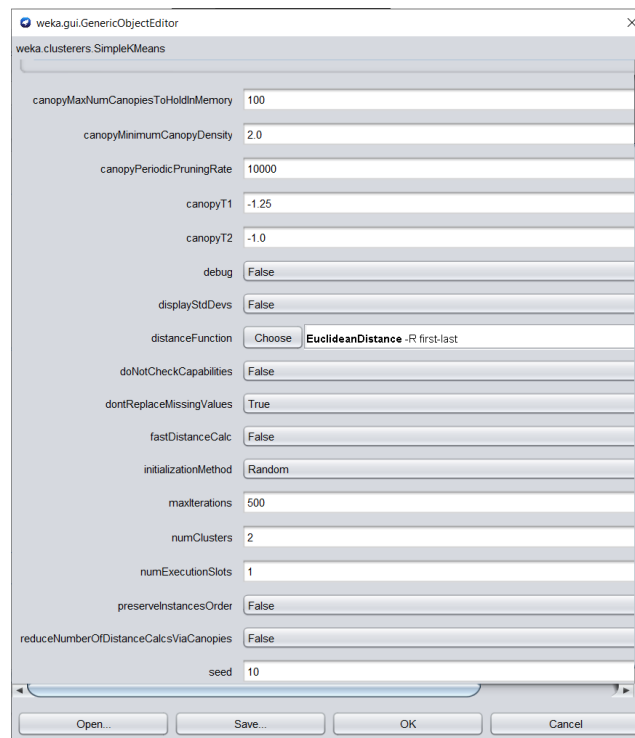
- 75% per l'addestramento dei dati.
- 25% per l'effettivo clustering.

Per applicare la tecnica clustering del *K-means*, ci si è recati al tab *Cluster* in alto, e, cliccando sul bottone *Choose*, si è scelto ***SimpleKMeans***:





Di seguito i parametri utilizzati (tutto a default, tranne *numClusters* = 2):



Di seguito, i risultati ottenuti:

Within cluster sum of squared errors: 159.76140156481802

Initial starting points (random):

Cluster 0: 34,1487680,81.4,19.8,46744,6.9,93.1,No

Cluster 1: 24,111888,80.3,23.1,38253,76.3,23.7,Yes

Final cluster centroids:

Attribute	Cluster#		
	Full Data	0	1
	(1077.0)	(679.0)	(398.0)
=====			
age	34.8143	36.8616	31.3216
population	397000.2646	282915.6259	591632.098
percent_completed_hs	83.8833	83.933	83.7985
poverty_rate	19.9773	18.9971	21.6497
median_income	47979.1291	48786.757	46601.2915
share_black	17.1613	10.11	29.191
share_not_black	82.8387	89.89	70.809
is_black	No	No	Yes

Time taken to build model (percentage split) : 0.01 seconds

Clustered Instances

```
0      221 ( 61%)
1      139 ( 39%)
```

L'applicazione del *K-means* ha elaborato due cluster, uno per classe, e uno di questi contiene il 61% dei data objects presenti nel dataset. In particolare, in questo cluster rientrano i record di classe *is\_black = No*, come ci si aspettava.

A fronte di un  $SSE = 159$ , ci si può ritenere soddisfatti dell'elaborazione.

I dettagli verranno approfonditi nel paragrafo del [paragrafo \*Evaluation\*](#).

## 6. Evaluation

In questo capitolo si discute la bontà dei risultati e, nel caso di più modelli proposti, ne segue la scelta di quello da offrire al cliente.

### 6.1 Prediction evaluation

Si procede ora alla visualizzazione di una tabella riassuntiva dei classificatori usati e dei risultati ottenuti, prima di procedere alla selezione:

Classifiers	Overall Accuracy	Precision	Recall	F-Measure	Time to build model
K-NN	76.8802 %	0.766	0.769	0.766	0 s
Classification tree	78.5515 %	0.784	0.786	0.785	0.03 s
Naive Bayes	77.1588 %	0.772	0.772	0.772	0 s
Ripper	77.9944 %	0.778	0.780	0.779	0.06 s

Alla luce dei risultati ottenuti a seguito della costruzione del dataset finale, tutti i classificatori possono essere valutati quasi alla stessa maniera, in quanto tutti si candidano come buoni modelli di classificazione per l'obiettivo da raggiungere.

Sebbene ci sia questa omogeneità, la scelta definitiva ricade sul modello dato dal *Classification tree* (in *Weka* il *J48*), in quanto, tra tutti, risulta, anche in minima parte, quello più performante.

### 6.2 Description evaluation

Il risultato del clustering ottenuto rispecchia quasi del tutto le ipotesi fatte [all'inizio di questo report](#).

I due cluster individuati, uno per classe, rappresentano quanto anticipato e riflettono quella che è la realtà statunitense, in particolare:

- Il primo cluster, indicato su *Weka* come **cluster 0**, rappresenta tutti quei data objects che indicano le vittime di etnia non afroamericana. I valori individuati, rappresentanti una media di tutte le istanze, indicano che, tendenzialmente, tutte le vittime avevano un'età media di quasi 37 anni, vivevano in una città più o meno popolosa (quasi trecentomila abitanti), in cui la percentuale di persone di etnia afroamericana è pari al 10%.
- Il secondo cluster, il **cluster 1**, rappresenta invece tutti i data objects che indicano le vittime di etnia afroamericana. I valori individuati, rappresentati da una media di tutte le istanze, indicano che, tendenzialmente, tutte le vittime avevano **un'età media di quasi 30 anni**, vivevano in una **città doppiamente popolosa** rispetto a quella individuata dal primo cluster (quasi **seicentomila abitanti**), in cui la **percentuale di persone di etnia afroamericana è pari al 29%** (quindi poco meno di un terzo della popolazione totale).

## 7. Deployment

### 7.1 Predictive task

Si è riusciti a costruire un modello con una *accuracy* del 78%. Sebbene non rispecchi una percentuale di affidabilità estremamente elevata, l'utilizzo di questo modello può rivelarsi uno strumento da tenere fedelmente in considerazione.

Avendo a disposizione una dead line più lontana, è sicuramente possibile migliorare ulteriormente le performance del modello, o addestrarne di altri. Sicuramente, gioverebbe anche l'utilizzo di un dataset più aggiornato, sia a livello di attributi e sia a livello di istanze, come anche un'analisi a propri più approfondita.

Il modello potrebbe essere utilizzato attraverso la costruzione di un software o di un'interfaccia grafica ad hoc.

### 7.2 Descriptive task

Si ipotizzava di trovare una divisione in cluster che potesse evidenziare un **racial bias** negli interventi della polizia statunitense.

Il lavoro svolto ha permesso di trovare due cluster ben definiti, uno dei quali rispetta quelle che sono state le ipotesi fornite a inizio report.

Le considerazioni fatte nel [paragrafo precedente](#) sono valide anche in questo caso.