
UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

# Traffic Violations

## Predizione delle violazioni del codice stradale

**Gruppo di lavoro**

Vito Giuseppe Natale, 697572, v.natale@studenti.uniba.it
Fiorente Domenico, 735556, d.fiorente1@studenti.uniba.it

**Repository**

https://github.com/FDomenico/ProgettoIcon.git

Ingegneria della conoscenza
AA 2023-24

# Sommario

# 1. Introduzione

La sicurezza stradale è un elemento cruciale per la gestione efficiente delle reti viarie e per la prevenzione degli incidenti. Le violazioni del codice stradale rappresentano un serio problema, influenzando la sicurezza degli utenti della strada e richiedendo risorse significative per il loro controllo e la loro gestione. In quest'ottica, l'impiego di approcci avanzati basati sull'ingegneria della conoscenza, in combinazione con l'intelligenza artificiale, rappresenta un passo significativo verso la prevenzione e la gestione ottimizzata delle violazioni del codice stradale.

Il caso di studio si propone di utilizzare metodologie di apprendimento automatico e di analisi dei dati per predire le violazioni del codice stradale in base a diverse variabili e parametri rilevanti. L'approccio si concentra sull'utilizzo di tecniche di classificazione avanzate per identificare i fattori determinanti associati alle violazioni e per sviluppare modelli predittivi in grado di prevedere tali comportamenti in contesti specifici.
In particolare, alla base del progetto vi è un dataset presente sul sito Kaggle che riporta i dati relativi alle violazioni commesse nell'America settentrionale.

Partendo dal dataset si è creata una base di conoscenza per consentire l'inferenza di nuove informazioni, successivamente utilizzate per testare e confrontare diversi modelli di machine learning.

# 2. Elenco argomenti di interesse

- Rappresentazione e ragionamento relazionale:
  - creazione e interrogazione di una knowledge base Prolog

- Apprendimento supervisionato:
  - modelli base di classificazione:
    - alberi di decisione
    - regressione logistica
    - support vector classifier
  - modelli ensable:
    - random forest
    - ada boost
  - case-base reasoning: k-nearest neighbors
  - naive Bayes
  - reti neurali
  - cross validation

- Apprendimento non supervisionato:
  - hard clustering: k-means

# 3. Descrizione e preprocessing del database

## 3.1 Sommario

Come anticipato nell'introduzione, si utilizza il dataset presente sul sito Kaggle che riporta i seguenti dati:

- **Description:** Descrizione testuale dell'infrazione specifica
- **Belts:** Se le cinture di sicurezza erano in uso nei casi di violazione o meno
- **Personal_Injury**: Se la violazione del traffico ha coinvolto lesioni personali o meno
- **Property_damaged**: Se la violazione del traffico ha coinvolto danni alla proprietà o meno
- **Commercial_license**: Se il conducente possiede una patente di guida commerciale o meno
- **Commercial_vehicle**: Se il veicolo che commette la violazione del traffico è un veicolo commerciale o meno
- **State**: Stato che emette la registrazione del veicolo
- **Vehicle_type**: Tipo di veicolo (Esempi: Automobile, Station Wagon, Heavy Truck, ecc.)
- **Years**: Anno di fabbricazione del veicolo
- **Make**: Produttore del veicolo (Esempi: Ford, Chevy, Honda, Toyota, ecc.)
- **Model**: Modello del veicolo
- **Color**: Colore del veicolo
- **Charge**: Codice alfanumerico per l'infrazione specifica
- **Contributed_to_accident**: Se la violazione del traffico è stata un fattore contributivo in un violazione o meno
- **Race**: Razza del conducente (Esempio: Asiatico, Nero, Bianco, Altro, ecc.)
- **Gender**: Genere del conducente (F = Femmina, M = Maschio)
- **Driver_city**: Città dell'indirizzo di casa del conducente
- **Driver_state**: Stato dell'indirizzo di casa del conducente
- **Dl_state**: Stato che emette la patente di guida
- **Arrest_type**: Tipo di arresto (A = Segnalato, B = Non segnalato, ecc.)
- **Violation_type**: Tipo di violazione (Esempi: Citation, Warning, SERO)

La nostra attenzione si è focalizzata sulla violation_type che può essere di tre tipi:

1. **Citation** (Contravvenzione), indica la violazione effettiva del codice stradale o delle leggi sul traffico, che potrebbe comportare multe o altre sanzioni. Solitamente, un avviso formale o un biglietto con l'indicazione dell'infrazione commessa.
2. **Sero,** è associata a violazioni legate a norme o regolamenti specifici del traffico o della circolazione stradale che riguardano l'applicazione di norme relative ai servizi di regolamentazione e controllo del traffico.
3. **Warning** (avvertimento), rappresenta una segnalazione da parte delle autorità di polizia riguardo a una possibile infrazione o violazione del codice stradale. Può essere un richiamo senza conseguenze dirette.

## 3.2 Strumenti utilizzati

Dato che il dataset è composto da file csv è stata utilizzata la libreria pandas che permette di leggere e manipolare dati memorizzati in questo tipo di file, oltre che fornire la possibilità di crearne di nuovi.

## 3.3 Decisioni di Progetto

Per quanto riguarda il preprocessing si è proceduto in questo modo:

1. **Aggiunta Unique_code**: vista la mancanza di un codice univo per distinguere le varie righe, si è ritenuto necessario inserirne uno.
2. **Conversione violation_type**: per rendere uniforme il dataset si è deciso di convertire la colonna `violation_type` da valori stringa in valori numerici (1 = Citation, 2 = SERO, 3 = Warning).
3. **Rimozione delle colonne**:
   - `'race'`: in quanto si è preferito tenere il campo 'gender'
   - `'state'`
   - `'driver_city'` si è preferito tenere il campo '`driver_state`' in quanto `driver_city` è meno influente avendo il campo '`driver_state`'.

4. **Eliminazione valori NULL**:
   Dato che saranno utilizzate per la creazione della knowledge base, sono stati eliminati valori nulli nelle colonne: ‘`belts`’, ‘`personal_injury`’, ‘`property_damaged`’, ‘`commercial_license`’, ‘`commercial_vehicle`’, ‘`vehicle_type`’, ‘`year`’, ‘`make`’, ‘`model`’, ‘`color`’, ‘`charge`’, ‘`contributed_to_accident`’, ‘`gender`’, ‘`driver_state`’, ‘`dl_state`’, ‘`arrest_type`’, ‘`violation_type`’.
5. **Rimozione righe con una sola parola**: nel campo description sono state eliminate quelle righe che contenevano un'unica parola come valore della descrizione poiché di difficile interpretazione.
6. **Classificazione e Tokenizzazione**: per poter utilizzare utilizzare il campo descrizione, è stato sottoposto ad un processo di tokenizzazione per ottenere la colonna keywords che è usata per il processo di classificazione per ottenere una categorizzazione della descrizione, ad es: `exceeding speed`, `driving alchool`, `careless driving`.
7. **Conversione category**: per rendere uniforme il dataset si è deciso di convertire la colonna `description_category` da valori stringa in valori numerici (4 = `exceeding speed`, 3 = `driving alchool`, 2 = `careless driving`, …).
8. **Formattazione colonna ‘year’**: Abbiamo deciso di formattare la colonna `year` contenente valori come 2012.0 in semplici valori come 2012 (togliendo il .0).

Infine, dato che il dataset risulta molto sbilanciato in quanto contiene molti pochi esempi in cui la label ‘`violation_type`’ è pari al 5% rispetto alle altre due, si effettua un bilanciamento del dataset, in modo da avere lo stesso numero di esempi per ogni label.

*Figura 1: Dataset prima del bilanciamento*

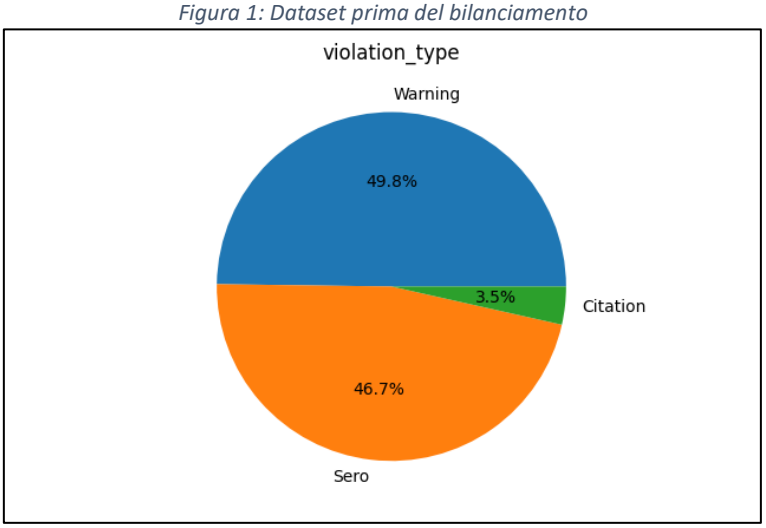
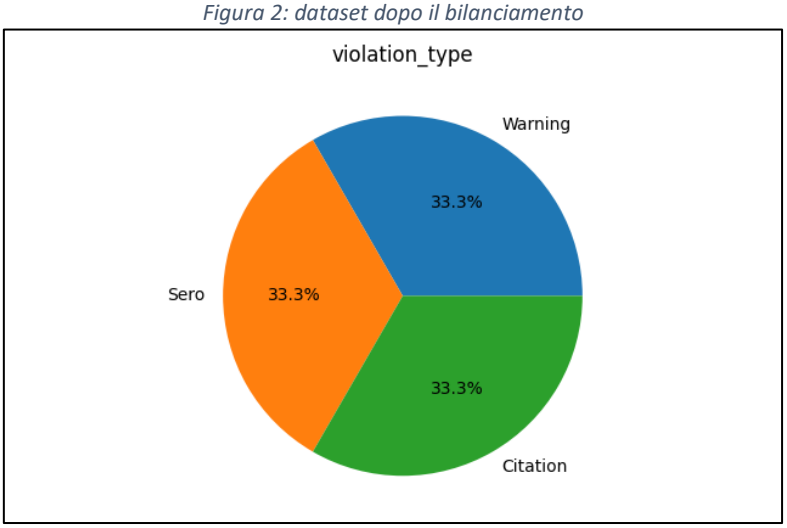



*Figura 2: dataset dopo il bilanciamento*



# 4. Valutazione

Dopo il preprocessing e il bilanciamento si è deciso di rappresentare il dataset utilizzando una base di conoscenza Prolog da interrogare per ottenere delle feature più facilmente utilizzabili nella fase di apprendimento automatico.

# 5. Creazione e interrogazione della knowledge base

## 5.1 Sommario

Partendo dal dataset ottenuto dopo il preprocessing, si è creata una base di conoscenza scritta in Prolog: si definiscono i fatti e le clausole e, successivamente, vengono poste delle query per creare il dataset su cui svolgere l'apprendimento.

## 5.2 Strumenti utilizzati

Per creare la knowledge base è stata utilizzata la libreria Python PySWIP la quale fornisce la possibilità di integrare la programmazione logica basata su Prolog nelle applicazioni Python. La libreria consente di creare, interrogare e manipolare conoscenze mediante regole e fatti logici.
Per creare il dataset ottenuto interrogando la KB è stata utilizzata la libreria pandas.

## 5.3 Decisioni di progetto

In questa sezione sono indicati i fatti e le clausole utilizzate per creare la knowledge base e, infine, sono indicati i campi che formano il dataset successivamente usato per l'apprendimento.

### Fatti

I fatti scritti nella kb sono i seguenti:

- `violation(Uc)`: violazione il cui `Unique_code` è `Uc`
- `category(violation(Uc),Ca)`: violazione il cui campo `category` è pari a `Ca`
- `belts(violation(Uc),B)`: violazione il cui campo `belts` è pari ad `B`
- `personal_injury(violation(Uc), PI)`: violazione il cui campo `personal_injury` è pari a `PI`
- `property_damage(violation(Uc), Pd)`: violazione il cui campo `property_damage` è pari a `Pd`
- `commercial_license(violation(Uc), Cl)`: violazione il cui campo `commercial_license` è pari a `Cl`
- `commercial_vehicle(violation(Uc), Cv)`: violazione il cui campo `commercial_vehicle` è pari a `Cv`
- `vehicle_type(violation(Uc), Vt)`: violazione il cui campo `vehicle_type` è `Vt`
- `year(violation(Uc), Y)`: violazione il cui campo `year` è pari a `Y`
- `make(violation(Uc), Ma)`: violazione il cui campo `make` è pari a `Ma`
- `model(violation(Uc), Mo)`: violazione il cui campo `model` è pari a `Mo`
- `color(violation(Uc), Co)`: violazione il cui campo `color` è `Co`
- `charge(violation(Uc), Ch)`: violazione il cui campo `charge` è pari a `Ch`
- `gender(violation(Uc), G)`: violazione il cui campo `gender` è pari a `G`
- `driver_state(violation(Uc), DS)`: violazione il cui campo `driver_state` è `Ds`
- `dl_state(violation(Uc), Dl)`: violazione il cui campo `dl_state` è pari a `Dl`
- `contributed_to_accident(violation(Uc), Cta)`: violazione il cui campo `contributed_to_accident` è `Cta`
- `arrest_type(violation(Uc), At)`: violazione il cui campo `arrest_type` è `At`

- violation_type(violation(Uc), Vt): violazione il cui campo violation_type è Vt

## Clausole definite

Le nostre clausole definite sono:

- current_year({current_year})
- is_belts(violation(V)) :- belts(violation(V1), B), (B = \"Yes\")
- is_commercial_licence(violation(V)) :- commercial_licence(violation(V), CL), (CL = \"Yes\")
- is_commercial_vehicle(violation(V)) :- commercial_vehicle(violation(V), CV), (CV = \"Yes\")
- is_contributed_to_accident(violation(V)) :- contributed_to_accident(violation(V), CTA), (CTA = \"Yes\")
- is_property_damaged(violation(V)) :- property_damage(violation(V), PD), (PD = \"Yes\")
- is_personal_injured(violation(V)) :- personal_injury(violation(V), PI), (PI = \"Yes\")

- is_m_patrol(violation(V)) :- arrest_type(violation(V), AT), (AT = \"A - Marked Patrol\")
- is_u_patrol(violation(V)) :- arrest_type(violation(V), AT), (AT = \"B - Unmarked Patrol\")
- is_l_p_recognition(violation(V)) :- arrest_type(violation(V), AT), (AT = \"S - License Plate Recognition\")
- is_m_laser(violation(V)) :- arrest_type(violation(V), AT), (AT = \"Q - Marked Laser\")
- is_motorcycle(violation(V)) :- arrest_type(violation(V), AT), (AT = \"L - Motorcycle\")
- is_f_patrol(violation(V)) :- arrest_type(violation(V), AT), (AT = \"O - Foot Patrol\")
- is_u_laser(violation(V)) :- arrest_type(violation(V), AT), (AT = \"R - Unmarked Laser\")
- is_marked(violation(V)) :- arrest_type(violation(V), AT), (AT = \"M - Marked (Off-Duty)\")
- is_m_s_radar(violation(V)) :- arrest_type(violation(V), AT), (AT = \"E - Marked Stationary Radar\")
- is_m_m_radar_s(violation(V)) :- arrest_type(violation(V), AT), (AT = \"G - Marked Moving Radar (Stationary)\")
- is_m_m_radar_m(violation(V)) :- arrest_type(violation(V), AT), (AT = \"I - Marked Moving Radar (Moving)\")
- is_u_m_radar_s(violation(V)) :- arrest_type(violation(V), AT), (AT = \"H - Unmarked Moving Radar (Stationary)\")
- is_monted_patrol(violation(V)) :- arrest_type(violation(V), AT), (AT = \"P - Mounted Patrol\")
- is_unmarked(violation(V)) :- arrest_type(violation(V), AT), (AT = \"N - Unmarked (Off-Duty)\")
- is_u_vascar(violation(V)) :- arrest_type(violation(V), AT), (AT = \"D - Unmarked VASCAR\")
- is_u_m_radar_m(violation(V)) :- arrest_type(violation(V), AT), (AT = \"J - Unmarked Moving Radar (Moving)\")
- is_u_s_radar(violation(V)) :- arrest_type(violation(V), AT), (AT = \"F - Unmarked Stationary Radar\")
- is_m_vascar(violation(V)) :- arrest_type(violation(V), AT), (AT = \"C - Marked VASCAR\")

- is_a_aircraft(violation(V)) :- arrest_type(violation(V), AT), (AT = \"K - Aircraft Assist\")

- arrest_category(violation(V), 1) :- is_m_patrol(violation(V)); is_u_patrol(violation(V)); is_f_patrol(violation(V)); is_monted_patrol(violation(V))")
- arrest_category(violation(V), 2) :- is_l_p_recognition(violation(V)); is_u_laser(violation(V)); is_m_laser(violation(V)); is_m_s_radar(violation(V)); is_m_m_radar_s(violation(V)); is_m_m_radar_m(violation(V)); is_u_m_radar_s(violation(V)); is_u_m_radar_m(violation(V)); is_u_s_radar(violation(V)); is_m_vascar(violation(V)); is_u_vascar(violation(V))")
- arrest_category(violation(V), 3) :- is_motorcycle(violation(V)); is_a_aircraft(violation(V)); is_unmarked(violation(V)); is_marked(violation(V))")

- vehicle_age(violation(V), Age) :- year(violation(V), Year), current_year(CurrentYear), Age is CurrentYear – Year)

- is_automobile(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"02 - Automobile\")
- is_l_truck(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"05 - Light Duty Truck\")
- is_other(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"28 - Other\")
- is_motorcycle(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"01 - Motorcycle\")
- is_r_vehicle(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"08 - Recreational Vehicle\")
- is_h_truck(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"06 - Heavy Duty Truck\")
- is_s_wagon(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"03 - Station Wagon\")
- is_moped(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"19 - Moped\")
- is_tractor(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"07 - Truck/Road Tractor\")
- is_u_trailer(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"25 - Utility Trailer\")
- is_t_bus(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"10 - Transit Bus\")
- is_c_rig(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"20 - Commercial Rig\")
- is_s_bus(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"12 - School Bus\")
- is_limousine(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"04 - Limousine\")
- is_farm_e(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"27 - Farm Equipment\")
- is_unknown(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"29 - Unknown\")
- is_ambulance(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"14 - Ambulance(Non-Emerg\")
- is_f_vehicle(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"09 - Farm Vehicle\")
- is_b_trailer(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"26 - Boat Trailer\")

- `is_t_trailer(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"21 - Tandem Trailer\")`
- `is_cc_bus(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"11 - Cross Country Bus\")`
- `is_camper(violation(V)) :- vehicle_type(violation(V), VT), (VT = \"24 - Camper\")`

- `vehicle_category(violation(V), 1) :- is_automobile(violation(V)); is_s_wagon(violation(V)); is_limousine(violation(V))`
- `vehicle_category(violation(V), 2) :- is_l_truck(violation(V)); is_h_truck(violation(V)); is_c_rig(violation(V)); is_tractor(violation(V))`
- `vehicle_category(violation(V), 3) :- is_r_vehicle(violation(V)); is_farm_e(violation(V)); is_camper(violation(V)); is_f_vehicle(violation(V)); is_ambulance(violation(V)); is_other(violation(V))`
- `vehicle_category(violation(V), 4) :- is_motorcycle(violation(V)); is_moped(violation(V))`
- `vehicle_category(violation(V), 5) :- is_t_bus(violation(V)); is_s_bus(violation(V)); is_cc_bus(violation(V))`
- `vehicle_category(violation(V), 6) :- is_unknown(violation(V))`
- `vehicle_category(violation(V), 7) :- is_b_trailer(violation(V)); is_t_trailer(violation(V)); is_u_trailer(violation(V))`

### Query

Per poter generare il dataset finale si pongono delle query alla base di conoscenza e i valori risultanti vengono salvati in un nuovo file csv utilizzato successivamente per l'apprendimento. Per evitare ripetizioni non si mostrano le query in quanto si rifanno alle due sezioni precedenti.
Il dataset ottenuto è formato delle seguenti features:

- `unique_code`: codice identificativo della violazione
- `violation_type`: indica il tipo di violazione con valori che vanno da 0 a 2 dove 0 indica 'citation', 1 indica 'SERO', 2 indica ' warning'
- `category`: indica la classe di identificazione per ciascuna violazione che è derivata dalla descrizione con valori che vanno da 0 a 16, alcuni esempi sono 4 = `exceeding speed`, 3 = `driving alchool`, 2 = `careless driving`, …
- `vehicle_age`: indica l'età del veicolo coinvolto nella violazione.
- `belts`: indica se il conducente ha la cintura o meno.
- `personal_injured`: indica se la violazione ha comportato danni alle persone o meno.
- `contributed_to_accident`: indica se la violazione è stato un fattore contributivo in un incidente o meno.
- `property_damaged`: indica se la violazione ha comportato danni alle cose o meno.
- `vehicle_category`: indica il tipo di veicolo con valori che vanno da 1 a 7, dove 1 sono le `car`, 2 sono i `truck`, 3 sono gli `other`, 4 sono gli `unknown`, 5 sono i `bus`, 6 sono i `trailer`, 7 sono i `motorcycle`.
- `arrest_category`: indica la categoria delle modalità di rilevamento delle infrazioni con valori che vanno da 1 a 3, dove 1 sono le `traffic`, 2 sono i `survey`, 3 sono gli `other1`.

# 6. Apprendimento supervisionato

## 6.1 Sommario

Partendo dal dataset ottenuto, sono stati testati diversi modelli di apprendimento supervisionato per cercare quello che riesca meglio a svolgere questo task di classificazione su tre classi.
Per ogni modello la feature obiettivo è `violation_type`, mentre le feature di input sono tutte le altre escluso `unique_code`.
Per testare ognuno dei modelli si è usata una 10-fold cross validation e i dati delle valutazioni sono ottenuti mediando le valutazioni di ognuno dei fold.
Inoltre, per ognuno dei metodi utilizzati sono state testate diverse combinazioni di alcuni degli iperparametri per cercare di trovare la migliore combinazione degli stessi, utilizzando l'accuracy come strumento di confronto.
Per ognuno dei modelli si mostra la valutazione media ottenuta dal k fold della combinazione degli iperparametri migliore, la matrice di confusione e, infine, l'accuracy ottenuta per ognuno dei fold.
Infine, prima dell'apprendimento, è stata utilizzato lo `StandardScaler` per scalare i valori delle feature in quanto alcune di queste hanno dei valori molto più grande delle altre, portando a dei vantaggi con il knn e svm.

## 6.2 Strumenti utilizzati

Si è utilizzata la libreria `scikit-Learn` per creare i modelli di classificazione utilizzati nel confronto. Inoltre, si è usata la libreria `scikeras` per creare ed addestrare la rete neurale.

## 6.3 Decisioni di progetto

In questa sezione si mostrano i modelli utilizzati per l'apprendimento insieme ai relativi iperparametri presi in considerazione e la combinazione di questi ultimi risultata migliore.

### 6.3.1 Decision tree

Si sono testate le diverse combinazioni dei seguenti iperparametri per il modello:

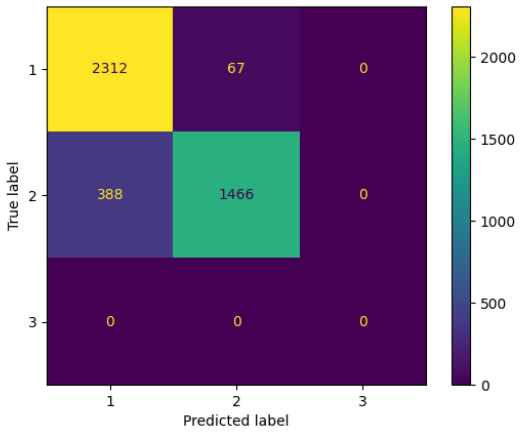
- `criterion`: misura la qualità della divisione all'interno dell'albero. I valori possibili sono:
  - `gini` [1]: valuta quanto un nodo sia "impuro" rispetto alle diverse classi dei dati. L'obiettivo è ridurre questo indice durante la suddivisione dei nodi per ottenere partizioni più omogenee e migliorare le previsioni del modello.
  - `entropy`: misura l'entropia nei nodi. Un'entropia migliore indica una divisione migliore
- `max_depth`: limita la profondità dell'albero. Sono stati testati i valori None, 5, 10, 20, 30
- `min_samples_split`: specifica il numero minimo di campioni richiesti in un nodo interno prima che possa essere suddiviso ulteriormente. Sono stati testati i valori 2, 5, 10, 20, 30
- `min_samples_leaf`: impone il numero minimo di campioni necessari in una foglia dell'albero. Sono stati testati i valori 1, 2, 5, 10, 20, 30

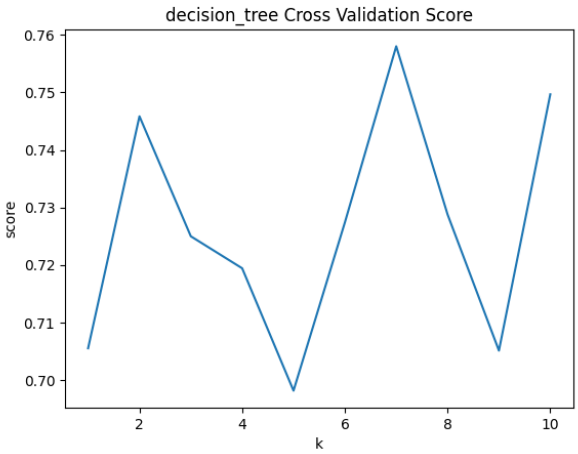
La migliore combinazione di iperparametri è stata:

- `criterion = entropy`
- `max_depth = 10`
- `min_samples_split = 30`
- `min_samples_leaf = 1`

Ha ottenuto i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.72 | 0.60 | 0.66 |
| **1** | 0.83 | 0.96 | 0.89 |
| **2** | 0.61 | 0.61 | 0.61 |
| **Macro avg** | 0.72 | 0.73 | 0.72 |
| **Accuracy** | 0.73 | | |



### 6.3.2 K-nearest neighbors

Si sono testate le diverse combinazioni dei seguenti iperparametri:

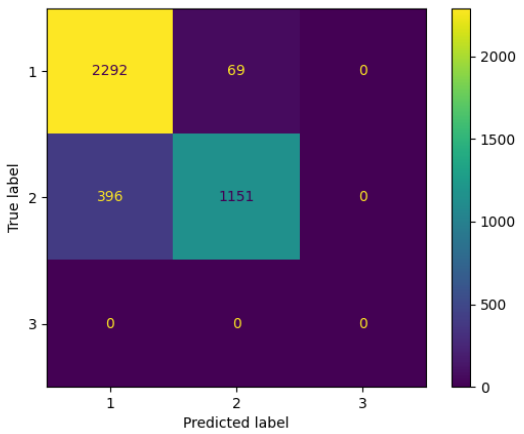
- `n_neighbors`: indica il numero di vicini da considerare quando si effettua una previsione. Sono stati testati i valori 3, 5, 7, 9, 11, 13, 15
- `weights`: specifica come assegnare pesi ai vicini quando si effettua la previsione. Le due possibilità sono:
  - `uniform`: tutti i vicini contribuiscono con lo stesso peso
  - `distance`: i vicini più vicini hanno un peso maggiore nella previsione
- `metric`: specifica la metrica per misurare la distanza tra i punti. Le opzioni possono essere:
  - `euclidean`: si calcola la radice quadrata della somma dei quadrati delle differenze tra le coordinate dei punti
  - `manhattan`: si calcola sommando le differenze assolute tra le loro coordinate lungo ciascuna dimensione
  - `chebyshev`: misura la massima differenza tra le coordinate di due punti in uno spazio multidimensionale calcolando la distanza lungo la dimensione in cui questa differenza è massima, ignorando le altre dimensioni

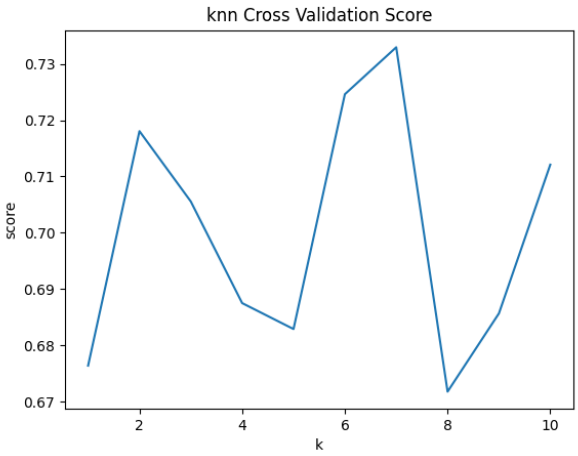
La migliore combinazione di iperparametri è stata:

- `n_neighbors = 15`
- `weights = distance`
- `metric = manhattan`

Ha ottenuto i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.64 | 0.66 | 0.65 |
| **1** | 0.82 | 0.96 | 0.88 |
| **2** | 0.60 | 0.48 | 0.53 |
| **Macro avg** | 0.69 | 0.70 | 0.69 |
| **Accuracy** | 0.70 | | |



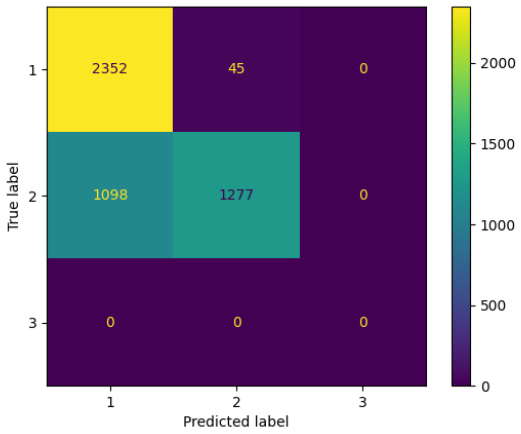



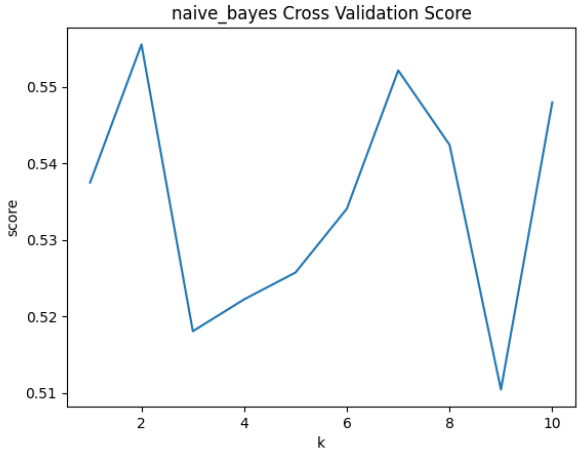
### 6.3.3 Naive bayes

Questo modello ha pochi iperparametri da testare, infatti l'unico testato è `var_smoothing` il quale aggiunge una quantità alle varianze delle features per evitare divisioni per zero durante il calcolo delle probabilità. I valori testati sono stati 0.1, 0.5, 1, 2, 5, 10.

Il valore per `var_smoothing` che si è rivelato migliore è stato 10 con i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.90 | 0.09 | 0.16 |
| **1** | 0.59 | 0.98 | 0.74 |
| **2** | 0.43 | 0.53 | 0.48 |
| **Macro avg** | 0.64 | 0.53 | 0.46 |
| **Accuracy** | 0.53 | | |



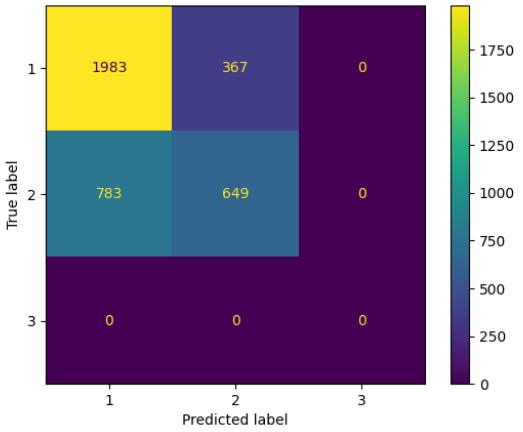



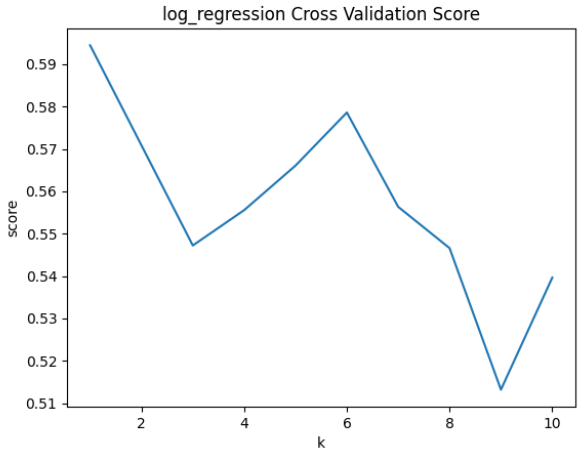
### 6.3.4 Logistic regression

Per la regressione logistica si è testato solo il parametro di regolarizzazione C: valori più bassi di C aumentano la regolarizzazione, mentre valori più alti di C la riducono. Sono stati testati i valori 0.1, 0.5, 1, 2, 5, 10.

Il valore del parametro C che ha portato al risultato migliore è stato 10:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.58 | 0.57 | 0.57 |
| **1** | 0.62 | 0.83 | 0.71 |
| **2** | 0.40 | 0.27 | 0.32 |
| **Macro avg** | 0.53 | 0.56 | 0.54 |
| **Accuracy** | 0.56 | | |



### 6.3.5 Support vector machine

Sono state testate le diverse combinazioni tra i seguenti iperparametri:

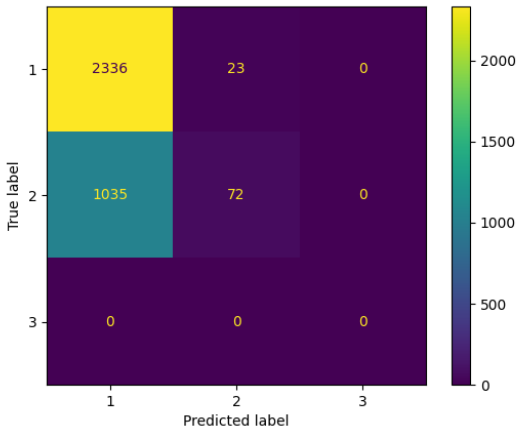
- `C`: controlla il trade-off tra la massimizzazione della larghezza della "margine" e la minimizzazione dell'errore di classificazione. I valori testati sono 0.1, 0.5, 1, 2, 5, 10.
- `kernel`: determina come vengono separate le classi:
  - `linear`: utilizza una trasformazione lineare per separare i dati
  - `poly`: utilizza una trasformazione polinomiale per separare i dati
  - `rbf`: utilizza una funzione radiale per separare i dati
  - `sigmoid`: utilizza una funzione sigmoide per separare i dati
- `gamma`: controlla la flessibilità del modello. Le due possibilità sono:
  - `scale`: usa una scala fissa per gamma
  - `auto`: calcola gamma in base alle dimensioni dei dati in ingresso

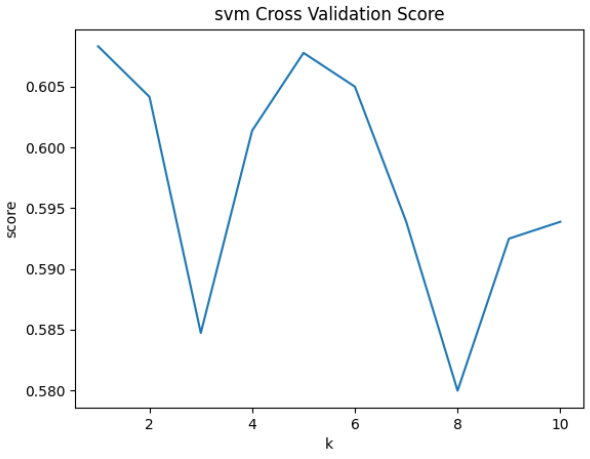
La migliore combinazione degli iperparametri è stata:

- `C = 1`
- `kernel = rbf`
- `gamma = scale`

Ha ottenuto i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.59 | 0.79 | 0.67 |
| **1** | 0.61 | 0.97 | 0.75 |
| **2** | 0.50 | 0.03 | 0.06 |
| **Macro avg** | 0.56 | 0.60 | 0.49 |
| **Accuracy** | 0.60 | | |



### 6.3.6 Random Forest

I parametri testati per questo modello sono:

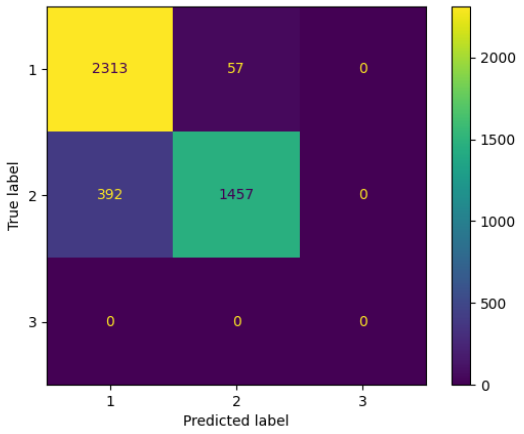
- `n_estimators`: indica quanti alberi decisionali verranno creati. Sono stati testati i valori 100, 200, 300
- `max_features`: controlla quanti attributi vengono presi in considerazione per ciascuna divisione in ciascun albero. I valori possibili sono:
  - `sqrt`: considera la radice quadrata del numero totale delle features
  - `log2`: considera il logaritmo in base due del numero totale delle features
- `max_depth`: limita la profondità di ogni albero. Si sono testati i valori None, 5, 10, 20, 30

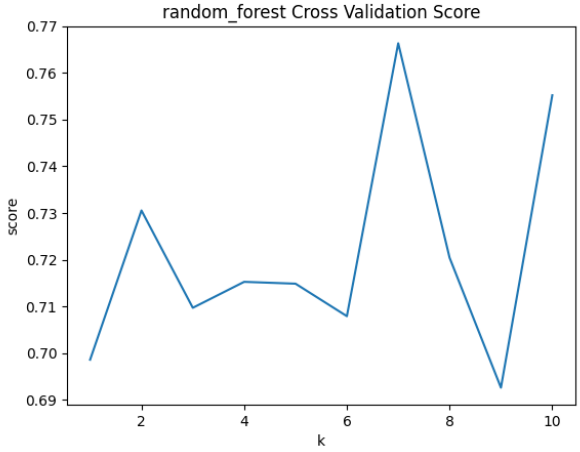
La combinazione migliore è stata:

- `n_estimators = 200`
- `max_features = log2`
- `max_depth = 10`

Ha ottenuto i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.71 | 0.59 | 0.65 |
| **1** | 0.83 | 0.96 | 0.89 |
| **2** | 0.61 | 0.61 | 0.61 |
| **Macro avg** | 0.72 | 0.72 | 0.71 |
| **Accuracy** | 0.72 | | |



### 6.3.7 Ada boost

Gli iperparametri testati sono:

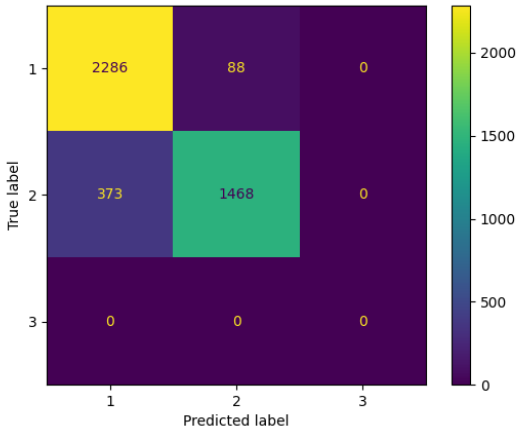
- `n_estimators`: indica quanti stimatori verranno creati. Sono stati testati i valori 50, 100, 200, 300
- `learning_rate`: controlla l'importanza di ciascuno stimatore nella combinazione finale. Sono stati presi in considerazione i valori 0.1, 0.5, 1, 2, 5, 10
- `estimator`: permette di specificare quale stimatore utilizzare. Sono stati testati: `DecisionTreeClassifier`, `GaussianNB`, `LogisticRegression`

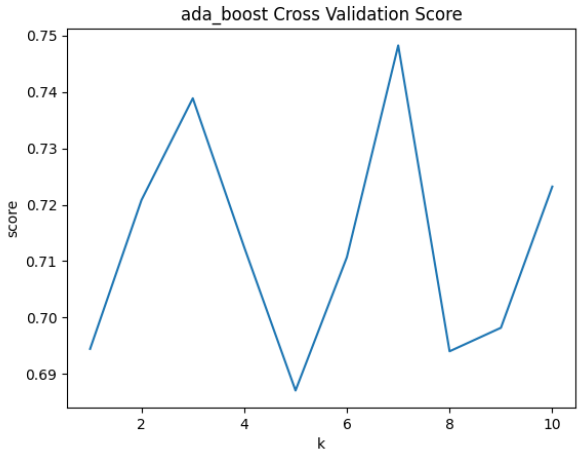
La migliore combinazione è data da:

- `n_estimators = 50`
- `learning_rate = 0.5`
- `estimator = DecisionTreeClassifier`

Ha ottenuto i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.70 | 0.58 | 0.63 |
| **1** | 0.83 | 0.95 | 0.89 |
| **2** | 0.59 | 0.61 | 0.60 |
| **Macro avg** | 0.71 | 0.71 | 0.71 |
| **Accuracy** | 0.71 | | |



### 6.3.8 Neural network

Si è creata una rete neurale sequenziale con questa topologia:

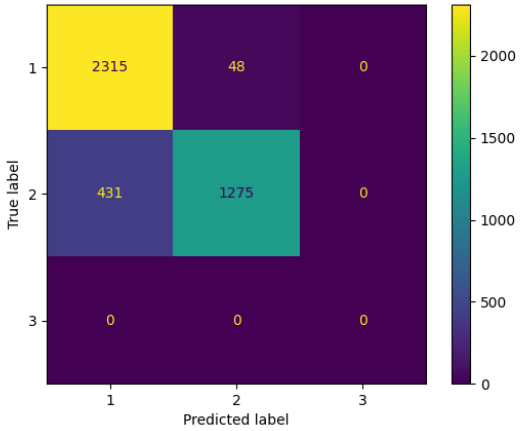
- un primo layer fully connected formato da `num_nodes` neuroni e funzione di attivazione `relu`
- viene applicato un dropout con probabilità `dropout_prob` per ridurre il rischio di overfitting tra i primi due strati
- un secondo layer fully connected con la metà dei nodi del primo e con funzione di attivazione `relu`
- uno strato finale formato da tre neuroni che rappresentano le classi di output, e utilizza l'attivazione `softmax` [2] per la classificazione multiclasse

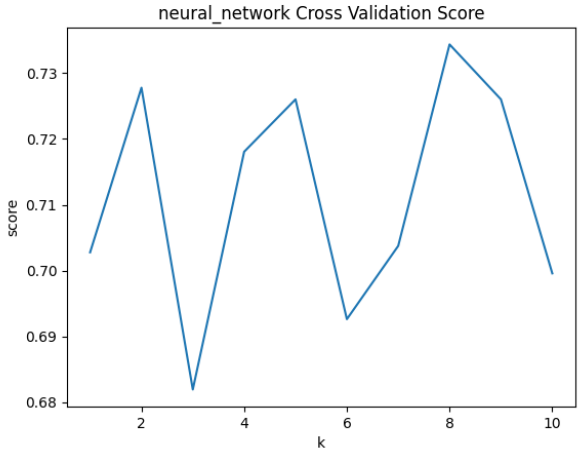
Sono state testate le diverse combinazioni dei seguenti parametri:

- `num_nodes`: definisce il numero di nodi che formeranno il primo, e di conseguenza, il secondo layer. Sono stati testati i valori 16, 32, 64, 128
- `epoch`: definisce per quante epoche deve essere addestrato il modello. Sono stati testati i valori 10, 50, 100
- `dropout_prob`: definisce la probabilità del dropout. Sono stati testati i valori 0, 0.2, 0.5

La rete che si è dimostrata migliore è stata quella formata da un primo strato di 128 nodi, il secondo da 64 nodi e probabilità di dropout al 20%, quando addestrato per 100 epoche, ottenendo i seguenti risultati:

| | Precision | Recall | F1 score |
|---|---|---|---|
| **0** | 0.68 | 0.66 | 0.67 |
| **1** | 0.81 | 0.97 | 0.88 |
| **2** | 0.62 | 0.53 | 0.57 |
| **Macro avg** | 0.71 | 0.72 | 0.71 |
| **Accuracy** | 0.72 | | |



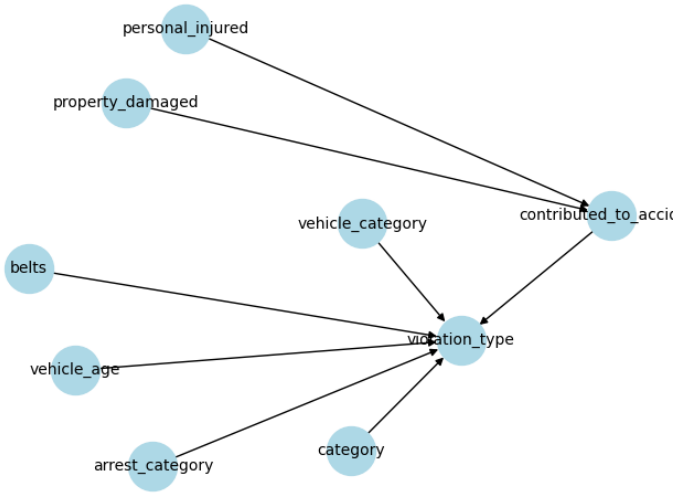



### 6.3.9 Bayesian network

Abbiamo creato un modello Bayesiano utilizzando la libreria `pgmpy`. Vengono importate le librerie necessarie e vengono letti i dati da un file CSV. I dati vengono quindi elaborati attraverso la discretizzazione utilizzando un algoritmo di `equal_width`. La discretizzazione consiste nel dividere i valori di ogni colonna in intervalli di larghezza uguale e assegnare ad ogni valore una lettera maiuscola (A, B, C, ecc.).
La funzione `val_counter` conta il numero di volte che ogni lettera maiuscola appare in una colonna specifica del DataFrame. La funzione `multi_val_counter` conta il numero di righe che soddisfano determinate condizioni su più colonne.
Le funzioni `generate_tuples` e `calc_matrix_with_cols` sono utilizzate per creare la matrice CPD (probabilità condizionata) per il modello Bayesiano. La matrice CPD descrive la probabilità di ottenere un determinato valore in una colonna in base ai valori in altre colonne.

```
model = BayesianNetwork(
    [
        ("category", "violation_type"),
        ("contributed_to_accident", "violation_type"),
        ("arrest_category", "violation_type"),
        ("vehicle_category", "violation_type"),
        ("vehicle_age", "violation_type"),
        ('belts', 'violation_type'),
        ("personal_injured", "contributed_to_accident"),
        ("property_damaged", "contributed_to_accident"),
    ]
)
```

Infine, il modello Bayesiano viene creato utilizzando la classe `BayesianNetwork` e la matrice CPD. La classe `VariableElimination` viene utilizzata per effettuare inferenze sul modello.




Fornisce una solida base per la creazione di un modello Bayesiano utilizzando la libreria pgmpy e per la sua applicazione a un dataset specifico.

```
expected_result: 2, 0,
actual_result: +------------------+----------------------+
| violation_type    |   phi(violation_type) |
+==================+======================+
| violation_type(0) |               0.1259 |
+------------------+----------------------+
| violation_type(1) |               0.6023 |
+------------------+----------------------+
| violation_type(2) |               0.2718 |
+------------------+----------------------+
expected_result: 2, 0,
actual_result: +------------------+----------------------+
| violation_type    |   phi(violation_type) |
+==================+======================+
| violation_type(0) |               0.1259 |
+------------------+----------------------+
| violation_type(1) |               0.6023 |
+------------------+----------------------+
| violation_type(2) |               0.2718 |
+------------------+----------------------+
Accuracy: 0.33337967468372026
```

## 6.4 Valutazione

Se si mettono insieme tutti i modelli in una tabella per confrontarne le prestazioni, considerando l'accuracy si può notare come ci sono lievi differenze tra i vari modelli:

| Modello | Accuracy |
|---|---|
| **Decision tree** | 0.73 |
| **Knn** | 0.70 |
| **Naive Bayes** | 0.53 |
| **Logistic regression** | 0.56 |
| **SVM** | 0.60 |
| **Random forest** | 0.72 |
| **Ada boost** | 0.71 |
| **Neural network** | 0.72 |
| **Bayesian network** | 0.33 |

Il modello che si è dimostrato leggermente migliore rispetto agli altri è stata il Decision tree, seguito da Random Forest e Neural network, ma nessuno di questi raggiunge uno score tale da essere significativo.
Anche le matrici di confusione confermano le lievi differenze tra i vari modelli.
Guardando le tabelle per ognuno dei modelli si nota come per la classe 2 (Warning) i valori di precision, recall e F1 sono sempre inferiori a quelli delle altre classi, nonostante le tre classi siano perfettamente bilanciate, dimostrando maggiore difficoltà nel riconoscere questa classe in particolare.
Inoltre, se si guardano i grafici dell'`accuracy` ottenuta per ognuno dei k-fold di tutti i modelli, si può osservare come, per quanto ci siano delle variazioni per il valore dell'`accuracy` queste sono lievi, dimostrando che ognuno dei modelli è risultato stabile e non è sensibile a variazioni casuali nei dati di allenamento e di test.
Diversamente dicasi per la rete bayesiana: infatti, l'accuratezza ottenuta (0.33) dall'esecuzione dell'algoritmo è decisamente inferiore alle altre.

# 7. Apprendimento non supervisionato

Partendo dalle stesse feature utilizzate per l'apprendimento supervisionato, si è provato a creare dei cluster utilizzando k-means. Prima di tutto, anche se si conosce il numero di classi presenti nel dataset, si è provato a vedere quale numero di cluster portasse ad una valutazione migliore usando:

- Inerzia [3]: misura la somma delle distanze quadrate tra ciascun punto dati e il centroide del suo cluster assegnato e l'obiettivo è quello di minimizzarla
- Silhouette Score [4]: misura quanto bene ogni punto dati è stato assegnato al suo cluster rispetto agli altri cluster. Il valore del Silhouette Score varia da -1 a 1, e un valore più alto indica un clustering migliore.

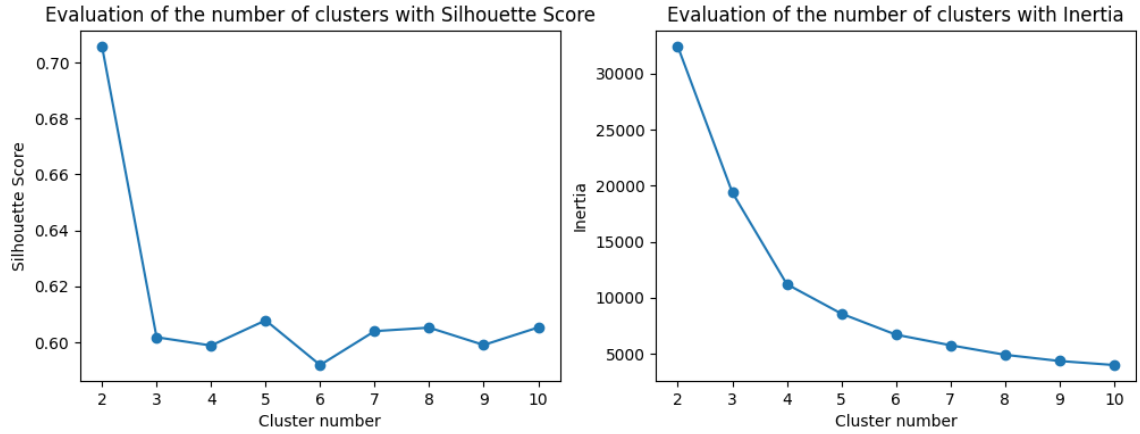



*Figura 3: Valutazione numero di cluster*

Si può notare come con un numero di cluster pari al numero di classi presenti nel dataset il valore di Silhouette Score è di circa 0.60 mentre l'inerzia è di circa 20000. Inoltre, si può vedere come all'aumentare del numero di cluster i valori di entrambe le metriche migliorano; infatti, il metodo del gomito suggerisce che potrebbe essere migliore suddividere il dataset in cinque classi invece che tre.

Dato che si è ottenuto questo risultato, si è provato ad utilizzare una metrica di valutazione esterna quando il numero di cluster è pari a tre. Nello specifico si è usato l'indice di Rand il quale varia da 0 a 1, dove un valore più alto indica una migliore concordanza tra le etichette reali e quelle del clustering.

Partendo dal numero di elementi in ciascuna classe si nota come non rispettano il dataset originale:

| | Etichette reali | Cluster |
|---|---|---|
| **0** | 2398 | 3006 |
| **1** | 2398 | 2455 |
| **2** | 2398 | 1733 |

Infatti, il valore ottenuto dall'indice di Rand è di 0.63, mostrando una piccola correlazione tra i dati reali e i cluster ottenuti, ma con un margine di miglioramento.

# 8. Conclusioni

Partendo dal dataset utilizzato nell'apprendimento nessuno dei modelli è riuscito ad ottenere delle prestazioni significative nel classificare le violazioni del codice stradale. Per cercare di migliorare la classificazione, si potrebbe pensare di ampliare le feature di partenza per poi estrarne delle altre che possano essere più significative nella classificazione, nel caso esistano, e successivamente testare i diversi modelli.
Una possibile estensione al progetto potrebbe essere quella di cercare di comprendere se è possibile dividere in dataset in cinque classi come suggerito dal metodo del gomito, anziché tre, e testare questa nuova suddivisione con i modelli utilizzati per vedere se porta ad una migliore classificazione.

# Riferimenti


[1] https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity.

[2] https://en.wikipedia.org/wiki/Softmax_function#Definition.

[3] https://www.codecademy.com/learn/machine-learning/modules/dspath-clustering/cheatsheet.

[4] https://en.wikipedia.org/wiki/Silhouette_(clustering).