ROAD SAFETY

predizione della gravità di un incidente

Progetto di:

Giuseppe Rubini 739073, g.rubini13@studenti.uniba.it

Repository:

https://github.com/PeppeRubini/Road-Safety

Ingegneria della conoscenza

AA 2022-23

Sommario

Introduzione

Gli incidenti stradali rappresentano una delle principali cause di morte e lesioni gravi in tutto il mondo. Ogni anno, milioni di persone subiscono gravi conseguenze a causa di incidenti stradali, con un impatto devastante sulle vite umane, sull'economia e sul sistema sanitario. Tuttavia, molte di queste tragedie potrebbero essere prevenute o mitigate attraverso un'analisi accurata e predittiva della gravità degli incidenti.

Il caso di studio si pone l'obiettivo di predire la gravità di un incidente stradale, in modo tale da informare i soccorritori e migliorare la gestione dei soccorsi, come, ad esempio, la quantità di ambulanze necessarie o quanto celere deve essere il soccorso per evitare fatalità.

In particolare, si è utilizzato un dataset fornito dal dipartimento dei trasporti del Regno Unito contenente gli incidenti avvenuti in territorio britannico, suddivisi per anno.

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

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

1. Descrizione e preprocessing del dataset

1.1 Sommario

Come anticipato nell'introduzione, si utilizza il dataset fornito dal dipartimento dei trasporti del Regno Unito relativo agli incidenti stradali. Nello specifico si sono presi in considerazione i file relativi agli anni da 2016 a 2021. Per ogni anno troviamo tre tipi di file contenenti informazioni diverse:

- **Accident**: contiene informazioni relative all'incidente, come la posizione, data e ora, condizioni meteo e del manto stradale. Particolarmente importanti sono il campo `'accident_index'` che identifica in modo univoco un incidente e `'accident_severity'` che assegna un valore da 1 a 3 all'incidente dove:
  - 1 = Fatal
  - 2 = Serious
  - 3 = Slight
- **Vehicle**: contiene informazioni generali riguardanti i veicoli coinvolti nell'incidente, come tipo di veicolo e l'età del veicolo. In particolare, il campo `'accident_index'` permette il join con i dati relativi all'incidente e il campo `'vehicle_reference'` che identifica un veicolo coinvolto in uno specifico incidente
- **Casualty**: contiene informazioni generali riguardo alle persone coinvolte nell'incidente. In modo simile al precedente c'è il campo `'accident_index'` per il join con l'incidente e il campo `'casualty_reference'` per identificare il ferito all'interno dell'incidente

La maggior parte dei campi del dataset sono interi, questo perché viene usata una codifica che mappa agli interi delle stringhe. Il significato di questi valori è spiegato nel file chiamato Road-Safety-Open-Dataset-Data-Guide.xlsx.

1.2 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.

1.3 Decisioni di Progetto

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

- **Accident**:

  1. rimosse le colonne:
     - `'accident_reference'` in quanto identifica un incidente solo all'interno dell'anno
     - `'latitude'`, `'longitude'`, `'location_easting_osgr'`, `'location_northing_osgr'`: forniscono le coordinate dell'incidente
     - `'accident_year'` in quanto c'è il campo `date` che è più generale
     - `'local_authority_district'` si è preferito il codice ONS
     - `'lsoa_of_accident_location'` considera un'area geografica troppo piccola

2. eliminati eventuali valori NULL, in quanto utilizzate per la creazione della knowledge base, nelle colonne: `'accident_index'`, `'accident_severity'`, `'number_of_vehicles'`, `'number_of_casualties'`, `'date'`, `'time'`, `'local_authority_ons_district'`, `'road_type'`, `'light_conditions'`, `'weather_conditions'`, `'road_surface_conditions'`, `'special_conditions_at_site'`, `'carriageway_hazards'`, `'first_road_class'`, `'first_road_number'`, `'second_road_class'`, `'second_road_number'`

3. eliminate quelle righe in cui ‘`road_type`’, ‘`first_road_number`’ o ‘`second_road_number`’ è pari a -1 in quanto questo valore significa missing

- **Vehicle**:

1. rimosse le colonne: `'accident_reference'`, `'accident_year'`, `'age_band_of_driver'`, `'driver_imd_decile'`, `'driver_home_area_type'` per motivi simili a quelli degli incidenti

2. eliminati eventuali valori NULL per: `'accident_index'`, `'vehicle_reference'`, `'vehicle_type'`, `'engine_capacity_cc'`, `'age_of_vehicle'`, `'propulsion_code'`, `'sex_of_driver'`, `'age_of_driver'`, `'towing_and_articulation'`, `'first_point_of_impact'`

- **Casualty**:

1. rimosse le colonne: `'accident_reference'`, `'accident_year'`, `'age_band_of_casualty'`, `'casualty_imd_decile'`, `'casualty_home_area_type'per` motivi simili ai precedenti

2. eliminati eventuali valori NULL per: `'accident_index'`, `'casualty_reference'`, `'casualty_severity'`, `'casualty_type'`

3. eliminate le righe in cui ‘`age_of_casualty`’ è pari a -1

Infine, dato che il dataset risulta molto sbilanciato in quanto contiene molti pochi esempi in cui la label ‘`accident_severity`’ è pari a 1 rispetto alle altre due, si effettua un bilanciamento del dataset, in modo da avere lo stesso numero di esempi per ogni label.

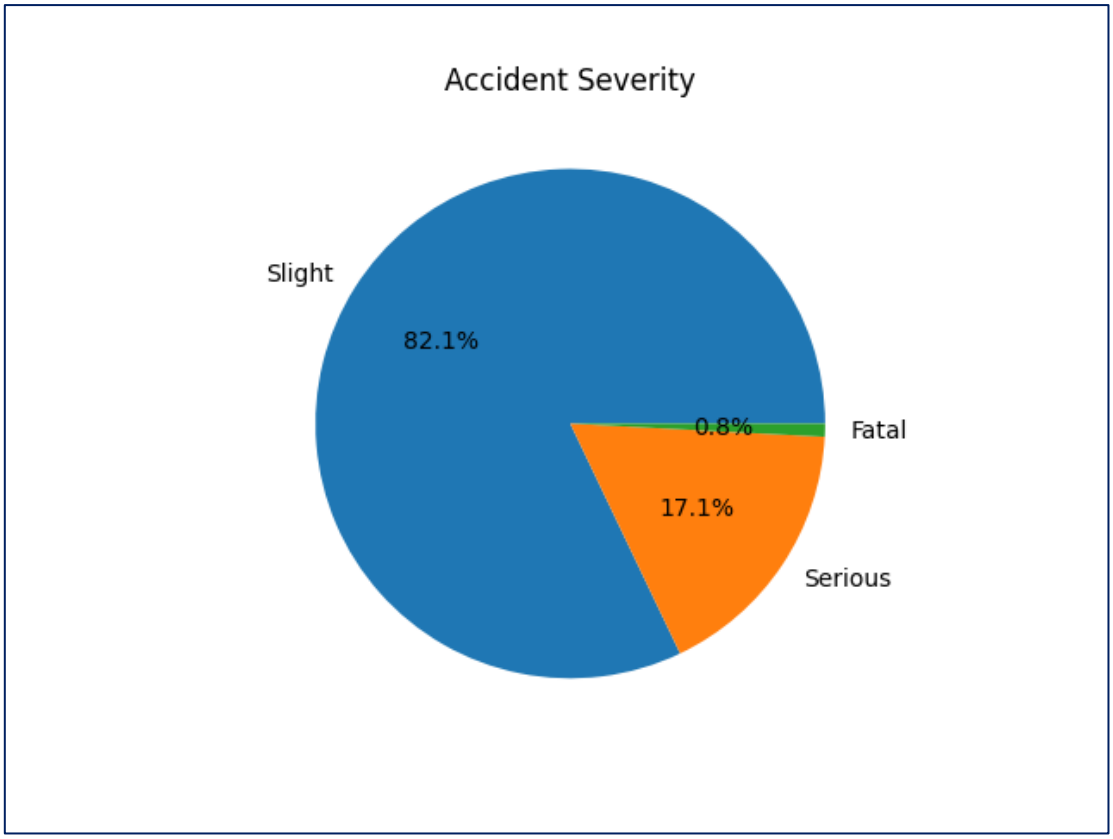


Figura 1: Dataset prima del bilanciamento

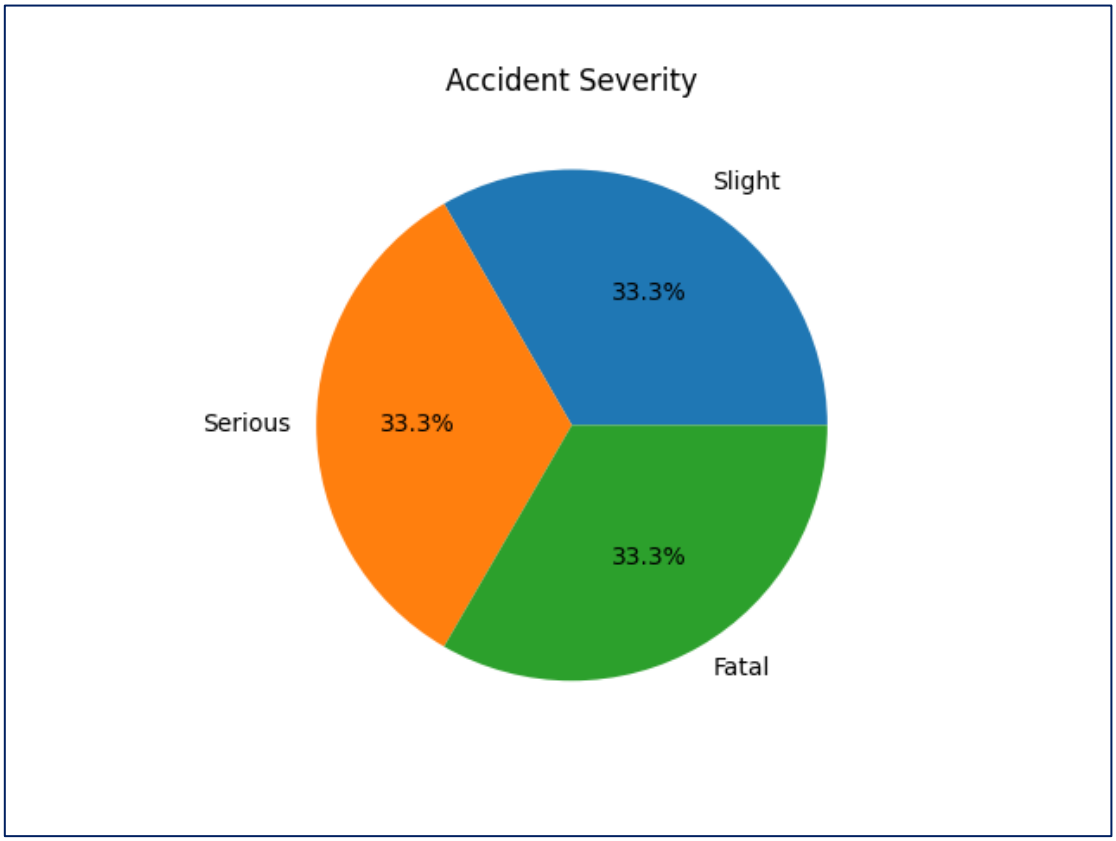


Figura 2: Dataset dopo il bilanciamento

1.4 Valutazione

Dopo il preprocessing e il bilanciamento, come si poteva intuire dal dominio preso in considerazione, si può notare che il numero medio di veicoli coinvolti in un incidente è 1.80 con un valore massimo di 9, evidenziano il fatto che c'è una relazione uno a molti tra incidente e veicoli. La stessa conclusione si può trarre per il numero di feriti che è di 1.38 con un valore massimo di 20.

Data la presenza di queste relazioni, 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.

2. Creazione e interrogazione della knowledge base

2.1 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.

2.2 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.

2.3 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.

2.3.1 Fatti

Si spiega solo il significato dei fatti relativi agli incidenti in quanto una volta capito come leggerli risulta semplice comprendere il significato dei fatti relativi ai veicoli e ai feriti. I fatti scritti nella kb sono i seguenti:

INCIDENTE

- `accident(A)`: incidente il cui accident_index è A
- `accident_severity(accident(A), AS)`: incidente il cui campo `accident_severity` è pari ad AS
- `number_vehicle(accident(A), NV)`: incidente il cui campo `number_of_vehicles` è pari a NV

- `number_casualty(accident(A), NC)`: incidente il cui campo `number_of_casualties` è pari a NC

- `date(accident(A), D)`: incidente il cui campo date è pari a D

- `time(accident(A), T)`: incidente il cui campo time è pari a T

- `accident_ons(accident(A), ONS)`: incidente il cui campo `local_authority_ons_district` è ONS

- `road_type(accident(A), RT)`: incidente il cui campo `road_type` è pari a RT

- `speed_limit(accident(A), SL)`: incidente il cui campo `speed_limit` è pari a SL

- `light_conditions(accident(A), LC)`: incidente il cui campo `light_conditions` è pari a LC

- `weather_conditions(accident(A), WC)`: incidente il cui campo `weather_conditions` è WC

- `road_surface_conditions(accident(A), RC)`: incidente il cui campo `road_surface_conditions` è pari a RC

- `special_conditions_at_site(accident(A), SC)`: incidente il cui campo `special_conditions_at_site` è pari a SC

- `carriageway_hazards(accident(A), H)`: incidente il cui campo `carriageway_hazards` è H

- `first_road_class(accident(A), FC)`: incidente il cui campo `first_road_class` è pari a FC

- `first_road_number(accident(A), FN)`: incidente il cui campo `first_road_number` è FN

- `second_road_class(accident(A), S)`: incidente il cui campo `second_road_class` è S

- `second_road_number(accident(A), SN)`: incidente il cui campo `second_road_number` è SN

- `pedestrian_crossing_facilities(accident(A), PC)`: incidente il cui campo `pedestrian_crossing_physical_facilities` è pari a PC

VEICOLO

- `vehicle(accident(A), V)`: veicolo coinvolto nell'incidente il cui `accident_index` è A con `vehicle_reference` V
- `vehicle_type(vehicle(accident(A), V), VT)`
- `engine_capacity(vehicle(accident(A), V), EC)`
- `propulsion_code(vehicle(accident(A), V), P)`
- `vehicle_age(vehicle(accident(A), V), VA)`
- `driver_sex(vehicle(accident(A), V), DS)`
- `driver_age(vehicle(accident(A), V), DA)`
- `towing_and_articulation(vehicle(accident(A), V), TA)`
- `first_point_of_impact(vehicle(accident(A), V), FI)`

FERITO

- `casualty(accident(A), C)`: ferito coinvolto nell'incidente il cui `accident_index` è A con `casualty_reference` V
- `casualty_sex(casualty(accident(A), C), CX)`
- `casualty_age(casualty(accident(A), C), CA)`
- `casualty_severity(casualty(accident(A), C), CS)`
- `casualty_type(casualty(accident(A), C), CT)`

2.3.2 Clausole definite

Per poter derivare nuove features sono state definite le seguenti clausole definite scritte in Prolog. Non verrà spiegato il significato di tutte le clausole in si ritiene che il loro significato sia abbastanza intuitivo. L'unico punto per cui può essere necessaria una spiegazione è la funzione **`findall`**. Questa funzione prende in input tre valori `findall(what, condition, list)`, dove:

- `what` è ciò che si desidera trovare
- `condition` è la condizione che deve essere soddisfatta
- `list` contiene tutti gli elementi che soddisfano la condizione

Dato che questa funzione viene utilizzata quando è necessario contare il numero di elementi, viene affiancata dalla funzione `length(List, Length)` dove `Length` è la variabile che conterrà la lunghezza della lista.

Le clausole sono le seguenti:

- `same_ons(accident(A1), accident(A2)) :- accident_ons(accident(A1), ONS), accident_ons(accident(A2), ONS)`
- `num_accident_in_ons(accident(A1), N) :- findall(A2, same_ons(accident(A1), accident(A2)), L), length(L, N)`
- `same_road_type(accident(A1), accident(A2)) :- road_type(accident(A1), RT), road_type(accident(A2), RT)`
- `num_accident_in_same_road_type(accident(A1), N) :- findall(A2, same_road_type(accident(A1), accident(A2)), L), length(L, N)`
- `same_road_class(accident(A1), accident(A2)) :- first_road_class(accident(A1), FC), first_road_class(accident(A2), FC)`
- `same_road_number(accident(A1), accident(A2)) :- first_road_number(accident(A1), FN), first_road_number(accident(A2), FN)`
- `same_road(accident(A1), accident(A2)) :- same_road_class(accident(A1), accident(A2)), same_road_number(accident(A1), accident(A2))`
- `num_accident_in_same_road(accident(A1), N) :- findall(A2, same_road(accident(A1), accident(A2)), L), length(L, N)`
- `same_second_road_class(accident(A1), accident(A2)) :- second_road_class(accident(A1), SC), second_road_class(accident(A2), SC)`
- `same_second_road_number(accident(A1), accident(A2)) :- second_road_number(accident(A1), SN), second_road_number(accident(A2), SN)`
- `same_second_road(accident(A1), accident(A2)) :- same_second_road_class(accident(A1), accident(A2)), same_second_road_number(accident(A1), accident(A2))`
- `num_accident_in_same_second_road(accident(A1), N) :- findall(A2, same_second_road(accident(A1), accident(A2)), L), length(L, N)`
- `is_night(accident(A)) :- time(accident(A), T), (T >= 21; T =< 4)`
- `is_casualty_child(casualty(accident(A), C)) :- casualty_age(casualty(accident(A), C), CA), CA =< 12`
- `is_casualty_old(casualty(accident(A), C)) :- casualty_age(casualty(accident(A), C), CA), CA >= 70`
- `is_pedestrian(casualty(accident(A), C)) :- casualty_type(casualty(accident(A), C), CT), CT == 0`
- `is_car(vehicle(accident(A), V)) :- vehicle_type(vehicle(accident(A), V), VT), (VT == 8; VT == 9; VT == 10; VT == 19)`
- `is_heavy_vehicle(vehicle(accident(A), V)) :- vehicle_type(vehicle(accident(A), V), VT), (VT == 11; VT == 20; VT == 21; VT == 98)`
- `is_motorcycle(vehicle(accident(A), V)) :- vehicle_type(vehicle(accident(A), V), VT), (VT == 2; VT == 3; VT == 4; VT == 5; VT == 22; VT == 23; VT == 97)`
- `is_pedal_cycle(vehicle(accident(A), V)) :- vehicle_type(vehicle(accident(A), V), VT), VT == 1`

- `num_cars(accident(A), N) :- findall(V, is_car(vehicle(accident(A), V)), L), length(L, N)`
- `num_heavy_vehicles(accident(A), N) :- findall(V, is_heavy_vehicle(vehicle(accident(A), V)), L), length(L, N)`
- `num_motorcycles(accident(A), N) :- findall(V, is_motorcycle(vehicle(accident(A), V)), L), length(L, N)`
- `num_pedal_cycles(accident(A), N) :- findall(V, is_pedal_cycle(vehicle(accident(A), V)), L), length(L, N)`

2.3.3 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:

- `accident_index`: codice identificativo dell'incidente
- `accident_severity`: indica la gravità dell'incidente con un valore da 1 a 3 dove 1 sta ad indicare la gravità maggiore
- `num_accident_in_ons`: numero di incidenti avvenuti nella stessa area geografica identificata dal codice ONS
- `num_accident_in_same_road_type`: numero di incidenti avvenuti sullo stesso tipo di strada
- `num_accident_in_same_road`: numero di incidenti avvenuti sulla stessa strada, indicata da nome e numero
- `num_casualties`: numero di vittime coinvolte nell'incidente
- `num_vehicle_involved`: numero di veicoli coinvolti nell'incidente
- `weather_conditions`: condizioni meteo al momento dell'incidente
- `light_conditions`: condizioni di luce al momento dell'incidente
- `road_surface_conditions`: condizioni del manto stradale al momento dell'incidente
- `carriageway_hazards`: pericoli sulla carreggiata al momento dell'incidente
- `special_conditions_at_site`: condizioni speciali sul luogo dell'incidente
- `is_night`: vale 1 se l'ora dell'incidente è compreso tra le 21 e le 4

- `num_cars`: indica il numero di automobili o simili coinvolte nell'incidente
- `num_heavy_vehicles`: indica il numero di mezzi pesanti coinvolti nell'incidente
- `num_motorcycles`: indica il numero di motocicli o simili coinvolti nell'incidente
- `num_pedal_cycles`: indica il numero di biciclette coinvolte nell'incidente
- `num_pedestrians`: indica il numero di pedoni coinvolti nell'incidente
- `is_child_involved`: vale 1 se almeno una delle persone coinvolte ha meno di 12 anni
- `is_old_involved`: vale 1 se almeno una delle persone coinvolte ha più di 70 anni

3. Apprendimento supervisionato

3.1 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 `accident_severity`, mentre le feature di input sono tutte le altre escluso `accident_index`.

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.

3.2 Strumenti utilizzati

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

3.3 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.

3.3.1 Decision tree

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

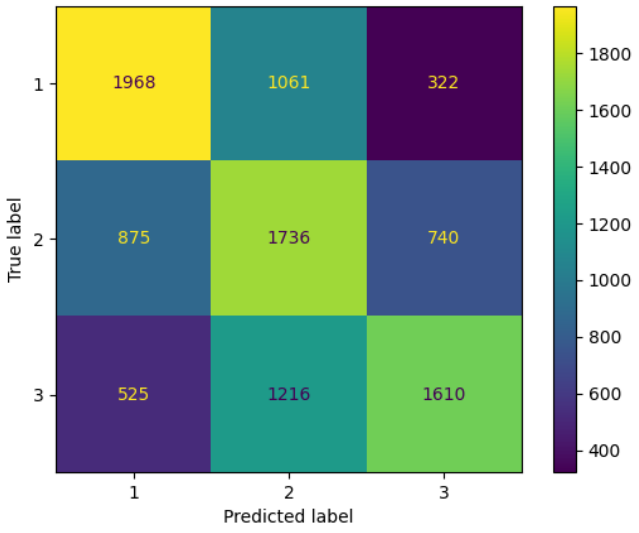
- `criterion`: misura la qualità della divisione all'interno dell'albero. I valori possibili sono:
  - `gini` [1]: valuta quanto un nodo sia "impuro" rispetto alle diverse classi dei dati. L'obiettivo è ridurre questo indice durante la suddivisione dei nodi per ottenere partizioni più omogenee e migliorare 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

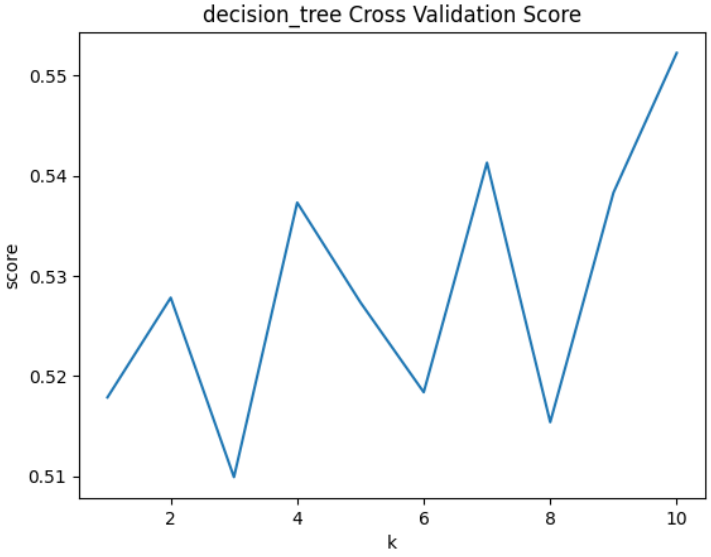
La migliore combinazione di iperparametri è stata:

- `criterion = gini`
- `max_depth = 10`
- `min_samples_split = 2`
- `min_samples_leaf = 20`

ed ha ottenuto i seguenti risultati:

	Precision	Recall	F1 score
1	0.58	0.59	0.59
2	0.43	0.52	0.47
3	0.60	0.48	0.53
Macro avg	0.54	0.53	0.53
Accuracy	**0.53**		





3.3.2 K-nearest neighbors

Si sono testate le diverse combinazioni dei seguenti iperparametri:

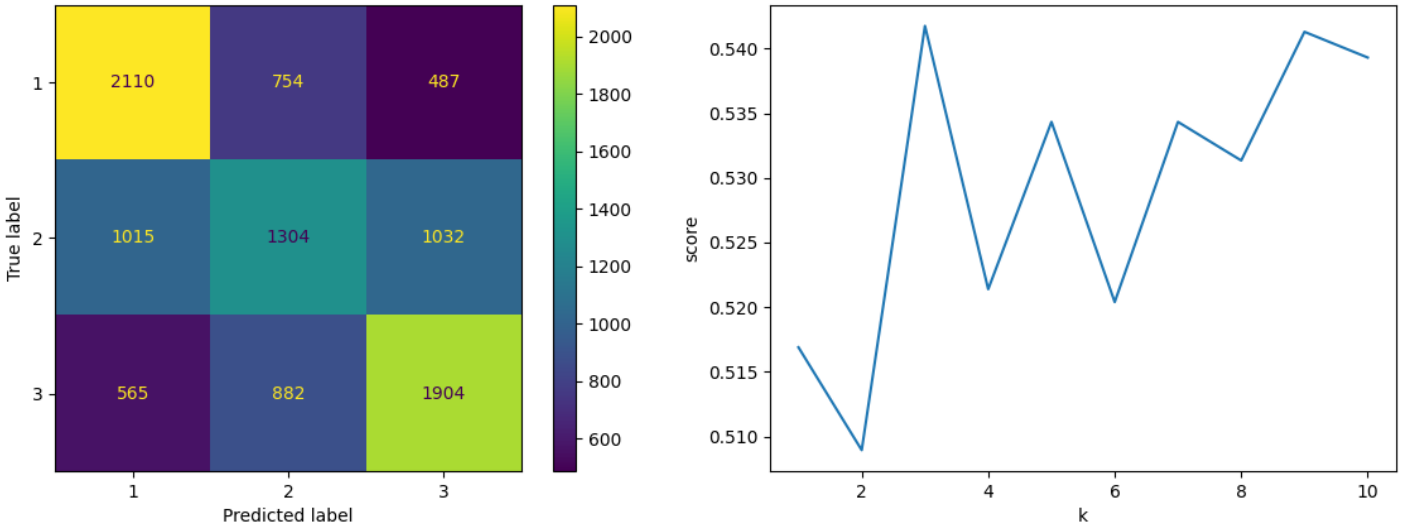
- `n_neighbors`: indica il numero di vicini da considerare quando si effettua una previsione. Sono stati testati i valori 3, 5, 7, 9, 11, 13, 15
- `weights`: specifica come assegnare pesi ai vicini quando si effettua la previsione. Le due possibilità sono:
  - `uniform`: tutti i vicini contribuiscono 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 = uniform`
- `metric = manhattan`

ed ha ottenuto i seguenti risultati:

	Precision	Recall	F1 score
1	0.57	0.63	0.60
2	0.44	0.39	0.41
3	0.56	0.57	0.56
Macro avg	0.52	0.53	0.53
Accuracy	**0.53**		

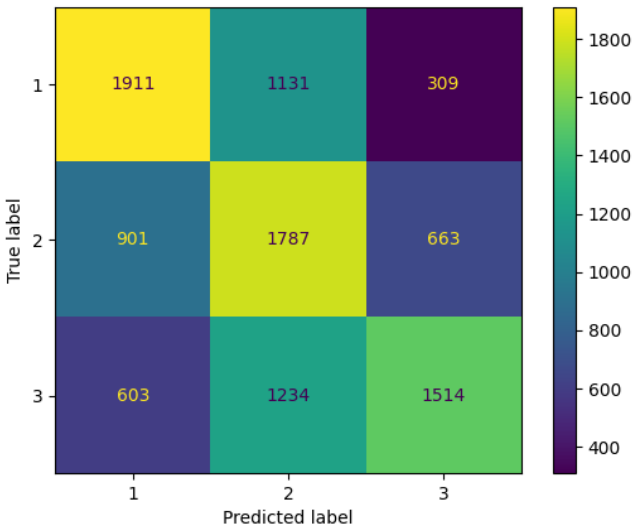


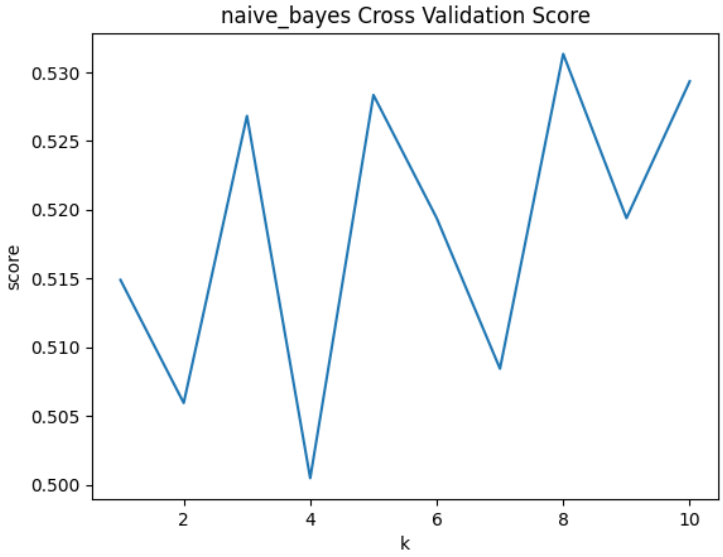
3.3.3 Naive bayes

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 2 con i seguenti risultati:

	Precision	Recall	F1 score
1	0.56	0.57	0.56
2	0.43	0.53	0.48
3	0.61	0.45	0.52
Macro avg	0.53	0.52	0.52
Accuracy	**0.52**		



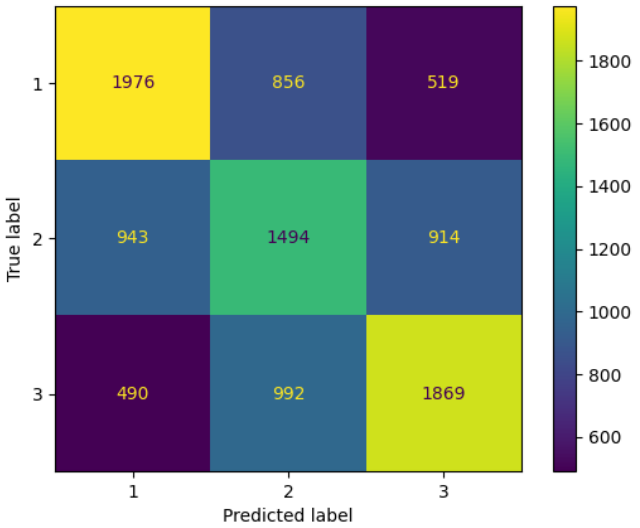


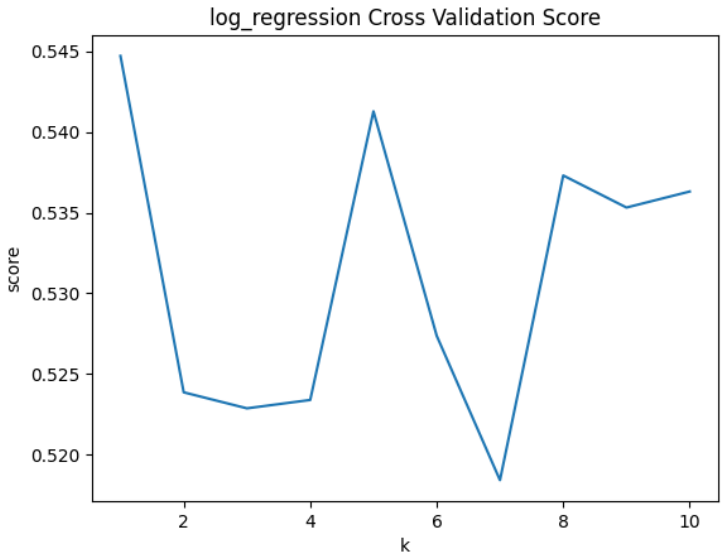
3.3.4 Logistic regression

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 0.1:

	Precision	Recall	F1 score
1	0.58	0.59	0.58
2	0.45	0.45	0.45
3	0.57	0.56	0.56
Macro avg	0.53	0.53	0.53
Accuracy	**0.53**		





3.3.5 Support vector machine

Sono state testate le diverse combinazioni tra i seguenti iperparametri:

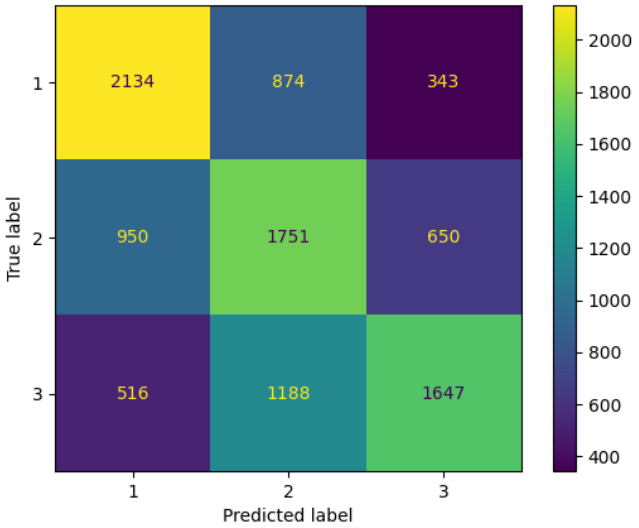
- `C`: controlla il trade-off tra la massimizzazione della larghezza della "margine" e la minimizzazione dell'errore di classificazione. I valori testati sono 0.1, 0.5, 1, 2, 5, 10.
- `kernel`: determina come vengono separate le classi:
  - `linear`: utilizza una trasformazione lineare per separare i 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

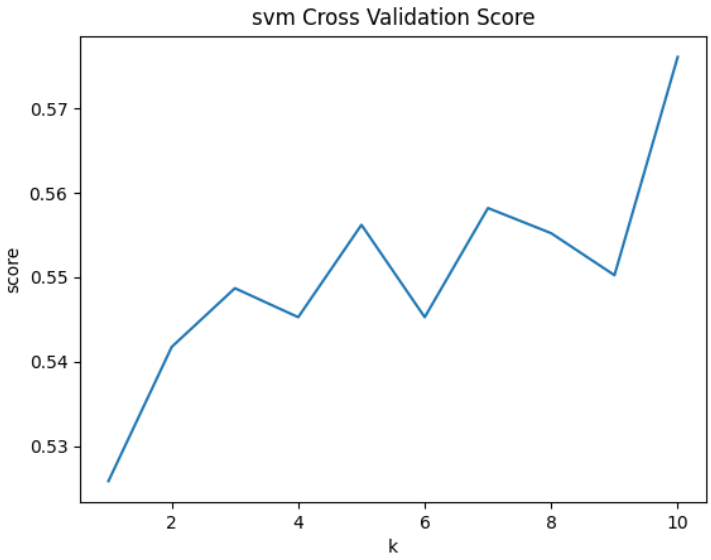
La migliore combinazione degli iperparametri è stata:

- `C = 0.5`
- `kernel = rbf`
- `gamma = scale`

ottenendo i seguenti risultati:

	Precision	Recall	F1 score
1	0.59	0.64	0.61
2	0.46	0.52	0.49
3	0.62	0.49	0.55
Macro avg	0.56	0.55	0.55
Accuracy	**0.55**		





3.3.6 Random forest

I parametri testati per questo modello sono:

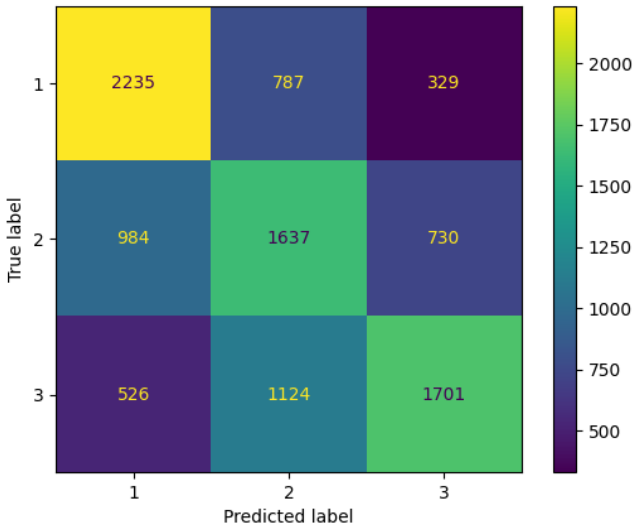
- `n_estimators`: indica quanti alberi decisionali verranno creati. Sono stati testati i valori 100, 200, 300
- `max_features`: controlla quanti attributi vengono presi in considerazione per ciascuna divisione in ciascun albero. I valori possibili sono:
  - `sqrt`: considera la radice quadrata del numero 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 = 300`
- `max_features = sqrt`
- `max_depth = 10`

ottenendo i seguenti risultati:

	Precision	Recall	F1 score
1	0.60	0.67	0.63
2	0.46	0.49	0.47
3	0.62	0.51	0.56
Macro avg	0.56	0.55	0.55
Accuracy	**0.55**		



3.3.7 Ada boost

Gli iperparametri testati sono:

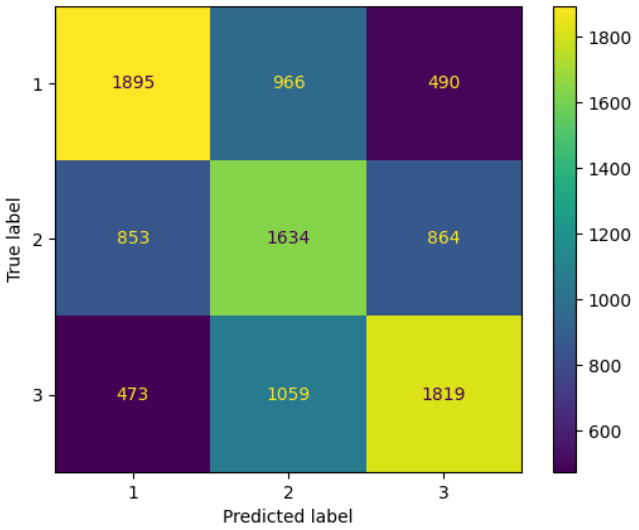
- `n_estimators`: indica quanti stimatori verranno creati. Sono stati testati i valori 50, 100, 200, 300
- `learning_rate`: controlla l'importanza di ciascuno stimatore nella combinazione finale. Sono stati presi in considerazione i valori 0.1, 0.5, 1, 2, 5, 10
- `estimator`: permette di specificare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

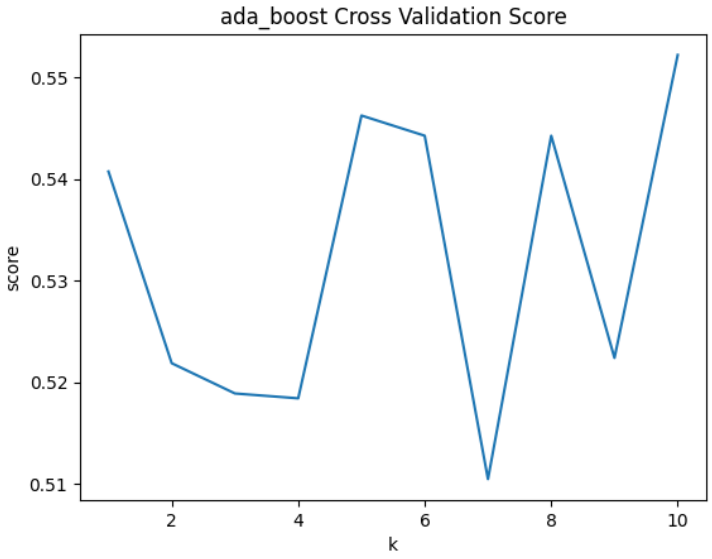
La migliore combinazione è data da:

- `n_estimators = 200`
- `learning_rate = 2`
- `estimator = LogisticRegression`

ottenendo i seguenti risultati:

	Precision	Recall	F1 score
1	0.59	0.57	0.58
2	0.45	0.49	0.47
3	0.57	0.54	0.56
Macro avg	0.54	0.53	0.53
Accuracy	**0.53**		





3.3.8 Neural network

Si è creata una rete neurale sequenziale con questa topologia:

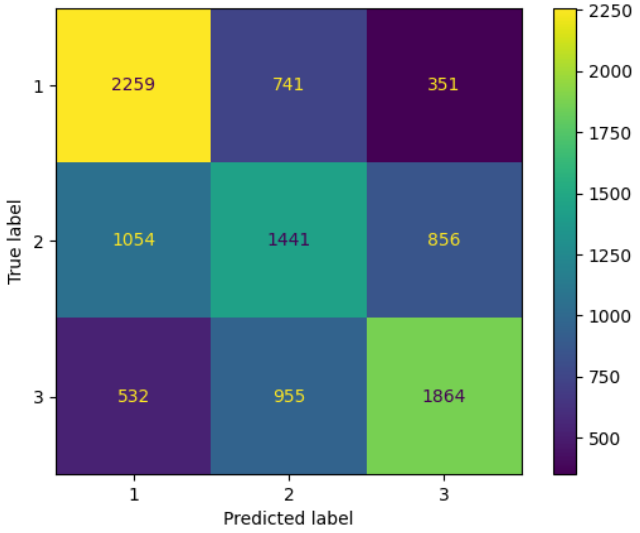
- un primo layer fully connected formato da `num_nodes` neuroni e funzione di attivazione `relu`
- viene applicato un dropout con probabilità `dropout_prob` per ridurre il rischio di overfitting tra i primi due strati
- un secondo layer fully connected con la metà dei nodi del primo e con funzione di 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

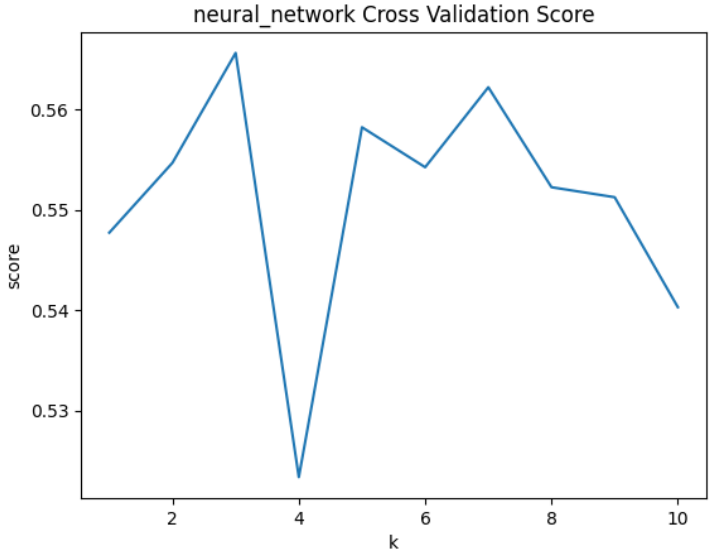
Sono state testate le diverse combinazioni dei seguenti parametri:

- `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 32 nodi, il secondo da 16 nodi e probabilità di dropout al 20%, quando addestrato per 50 epoche, ottenendo i seguenti risultati:

	Precision	Recall	F1 score
1	0.60	0.67	0.63
2	0.47	0.46	0.46
3	0.61	0.54	0.57
Macro avg	0.56	0.56	0.56
Accuracy	**0.56**		





3.4 Valutazione

Se si mettono insieme tutti i modelli in una tabella per confrontarne le prestazioni, considerando l'accuracy si può notare come tutti i modelli hanno ottenuto un risultato simile:

Modello	Accuracy
Decision tree	0.53
Knn	0.53
Naive Bayes	0.52
Logistic regression	0.53
SVM	0.55
Random forest	0.55
Ada boost	0.53
Neural network	0.56

Il modello che si è dimostrato leggermente migliore rispetto agli altri è stata la rete neurale, seguita da SVM e random forest, ma nessuno di questi raggiunge uno score tale da essere significativo.

Anche le matrici di confusione confermano come i risultati ottenuti dai modelli sono molto simili tra loro.

Guardando le tabelle per ognuno dei modelli si nota come per la classe 2 (Serious) 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.

4. 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.

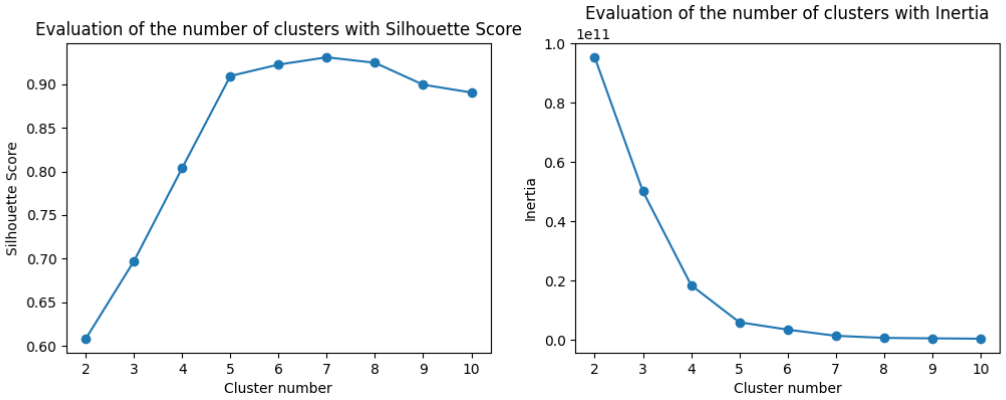


Figura 3: Valutazione numero di cluster

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 0.7 mentre l'inerzia è di circa 5. Inoltre, si può vedere come all'aumentare del numero di cluster i valori di entrambe le metriche; 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
1	3351	5820
2	3351	2564
3	3351	1669

Infatti, il valore ottenuto dall'indice di Rand è di 0.52, mostrando una piccola correlazione tra i dati reali e i cluster ottenuti, ma con un margine di miglioramento.

5. Conclusioni

Partendo dal dataset utilizzato nell'apprendimento nessuno dei modelli è riuscito ad ottenere delle prestazioni significative nel classificare la gravità degli incidenti. Per cercare di migliorare la classificazione, si potrebbe pensare di estrarre delle altre feature 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.

Un'altra estensione al progetto, una volta trovato un modello con una buona accuracy, può essere quella di creare un'interfaccia grafica che permetta ad un utente di inserire i dati necessari ed utilizzare il modello addestrato per effettuare la predizione.

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).