Pianificazione Urbana

Gruppo di lavoro

- Gaetano Malerba – Matricola: 726238

  g.malerba11@studenti.uniba.it
- Enrico Mangia – Matricola: 720678

  e.mangia4@studenti.uniba.it

Link repository GitHub:

https://github.com/GaeMale/Pianificazione_urbana

AA 2024-2025

1 Introduzione

Nel contesto della crescente urbanizzazione e dell'aumento del traffico veicolare, la sicurezza stradale rappresenta una sfida critica per le amministrazioni comunali di tutto il mondo. Incidenti stradali non solo causano perdite umane e feriti gravi, ma generano anche costi sociali ed economici significativi. Il nostro progetto nasce con l'obiettivo ambizioso di affrontare questa problematica attraverso un approccio innovativo basato sull'**analisi avanzata dei dati e sull'intelligenza artificiale**.

Cosa Fa il Nostro Progetto?

Il cuore del nostro sistema è un modello predittivo e di supporto alle decisioni, che analizza un'ampia gamma di dati relativi alla rete stradale urbana. Integrando informazioni provenienti da diverse fonti, il sistema è in grado di:

1. **Identificare i Punti Critici (Hotspot di Rischio):** Utilizzando dati reali (o, nel nostro caso, fittizi ma in modo accurato) e generati di incidenti stradali e flussi di traffico, il modello calcola un "indice di rischio" per ogni intersezione e segmento stradale all'interno di un'area urbana specifica (es. Bari, Terlizzi). Questo indice non si basa solo sulla storia degli incidenti, ma anche su fattori strutturali e ambientali che contribuiscono alla probabilità di un evento futuro.
2. **Prevedere la Probabilità di Incidenti:** Attraverso algoritmi di Machine Learning (come il Random Forest Classifier), il sistema è in grado di stimare la probabilità che un incidente si verifichi in un dato nodo della rete stradale, fornendo una visione predittiva del rischio.
3. **Generare Raccomandazioni di Intervento Specifiche:** Sulla base dell'indice di rischio calcolato e delle caratteristiche uniche di ogni punto critico (numero e gravità degli incidenti passati, volume di traffico, tipologia di strada, presenza di POI come scuole o semafori), il sistema propone raccomandazioni concrete e prioritarie per interventi di miglioramento della sicurezza stradale. Queste raccomandazioni variano da piccole migliorie alla segnaletica e all'illuminazione, fino a interventi strutturali complessi come la riprogettazione di incroci.
4. **Simulare Dati di Traffico e Incidenti (per lo sviluppo):** Per ovviare alla potenziale scarsità di dati reali dettagliati, il progetto include un robusto modulo di generazione di dati fittizi ma realistici, consentendo lo sviluppo, il testing e la dimostrazione del modello anche in assenza di dataset completi.

A Cosa Serve e Qual è il Suo Valore Aggiunto?

Il nostro progetto fornisce uno strumento potenzialmente strategico per le pubbliche amministrazioni, i pianificatori urbani e le forze dell'ordine, permettendo loro di effettuare:

- **Pianificazione Proattiva della Sicurezza:** Transitare da un approccio reattivo (intervenire dopo che gli incidenti si sono verificati) a uno proattivo (identificare e mitigare i rischi prima che si trasformino in incidenti).
- **Ottimizzazione delle Risorse:** Indirizzare gli investimenti e le risorse economiche limitate verso gli interventi più efficaci e urgenti, massimizzando l'impatto sulla sicurezza stradale.
- **Supporto alle Decisioni Basato sui Dati:** Fornire un fondamento oggettivo e analitico per le decisioni di pianificazione urbana, superando le valutazioni basate unicamente sull'esperienza o sull'intuizione.
- **Migliorare la Qualità della Vita:** Contribuire significativamente alla riduzione degli incidenti, salvando vite umane, diminuendo i feriti e migliorando la percezione generale di sicurezza per cittadini, pedoni e ciclisti.

In definitiva, il nostro progetto mira a essere un alleato tecnologico per costruire città più sicure e resilienti, trasformando i dati in azioni concrete per la protezione della comunità.

2 Requisiti funzionali

Il progetto è interamente sviluppato in **Python**. La scelta di Python è stata dettata dalla sua versatilità, dalla ricchezza del suo ecosistema di librerie per l'analisi dati, il machine learning e la manipolazione geospaziale, nonché dalla sua leggibilità, che facilita la collaborazione e la manutenibilità del codice.

2.1 Ambiente di Sviluppo Integrato (IDE)

L'IDE primario utilizzato per lo sviluppo è **IntelliJ IDEA**, con il plugin Python abilitato. IntelliJ IDEA è stato scelto per le sue avanzate funzionalità di:

- **Refactoring:** Strumenti intelligenti per la ristrutturazione del codice.
- **Debugging:** Potenti capacità di debug per l'identificazione e la risoluzione rapida dei problemi.
- **Code Completion:** Completamento automatico del codice contestuale per aumentare la velocità di sviluppo.
- **Integrazione con Version Control Systems:** Supporto nativo per Git, facilitando la gestione del controllo versione.
- **Gestione degli Ambienti Virtuali:** Funzionalità integrate per la creazione e la gestione di ambienti virtuali (venv), garantendo l'isolamento delle dipendenze del progetto.

2.2 Librerie utilizzate

Per l'implementazione dei diversi moduli e delle funzionalità richieste, il progetto fa ampio uso delle seguenti librerie Python, tra le più consolidate nell'ambito della data science e dell'analisi geospaziale:

- **Pandas:** Per la manipolazione e l'analisi dei dati tabulari, gestione di DataFrame efficienti.
- **NumPy:** Fondamentale per operazioni numeriche e computazioni scientifiche ad alte prestazioni.
- **Osmnx:** Essenziale per l'interazione con i dati di OpenStreetMap (OSM), consentendo il download, la modellazione, l'analisi e la visualizzazione delle reti stradali.
- **Scikit-learn:** Per l'implementazione degli algoritmi di Machine Learning (es. Random Forest Classifier) per la predizione del rischio di incidenti e la gestione delle metriche di valutazione.
- **GeoPandas:** Estensione di Pandas per la gestione di dati geospaziali, facilitando operazioni su geometrie e sistemi di coordinate.
- **OpenPyXL:** Necessario per la lettura e scrittura di file Excel (.xlsx), utilizzato per l'importazione di dati strutturati e l'esportazione dei risultati.
- **Matplotlib / Seaborn:** Per la visualizzazione dei dati, la creazione di grafici e l'esplorazione delle distribuzioni.
- **owlready2**: modulo Python che funge da ponte tra il mondo delle ontologie basate su OWL/RDF e il mondo della programmazione Python.

2.3 Gestione delle Dipendenze e Ambienti Virtuali

Tutte le dipendenze del progetto sono gestite tramite **ambienti virtuali (venv)**. Questo garantisce che:

- Le librerie siano isolate per ogni progetto, prevenendo conflitti di versione.
- L'ambiente di sviluppo sia pulito e le dipendenze siano chiaramente definite nel file *requirements.txt*.
- La portabilità del progetto sia massima, consentendo a qualsiasi sviluppatore di ricreare l'ambiente esatto e riprodurre i risultati.

2.4 Requisiti Funzionali Specifici del Sistema

Il sistema è progettato per supportare le seguenti funzionalità principali:

1. **Acquisizione Dati Geospaziali:** Capacità di scaricare ed elaborare dati di rete stradale da OpenStreetMap per un'area geografica specificata.
2. **Integrazione e Preprocessing Dati:** Moduli per l'integrazione di dati di incidenti, punti di interesse (POI) e conteggi di traffico con la rete stradale OSM, e per la pulizia, normalizzazione e ingegneria delle feature.
3. **Modellazione Predittiva del Rischio:** Implementazione di un modello di Machine Learning per la predizione del rischio di incidenti basato sulle feature estratte.
4. **Generazione dell'Indice di Rischio:** Calcolo di un indice di rischio composito per ogni nodo/segmento della rete, aggregando probabilità predette, dati storici di incidenti e altre feature.
5. **Raccomandazioni di Intervento:** Produzione di raccomandazioni testuali e prioritarie per interventi di sicurezza stradale, basate sull'indice di rischio e sulla logica decisionale definita.
6. **Simulazione Dati:** Funzionalità per la generazione di dataset di incidenti e traffico sintetici, ma statisticamente rappresentativi, per testare e dimostrare le capacità del modello in scenari variabili.
7. **Output dei Risultati:** Capacità di esportare i dati elaborati, gli indici di rischio e le raccomandazioni in formati strutturati (es. DataFrame Pandas, file Excel) per ulteriore analisi o integrazione con altri sistemi.
8. **Interazione/Interrogazione KB:** Permette di interagire con la KB mediante apposito menù.

2.5 Installazione e avvio

1. **Prerequisiti:** Prima di iniziare, assicurati di avere installati sul tuo sistema:

   - **Python 3.9 o superiore:** Si raccomanda una versione recente di Python 3. (Puoi scaricarlo da python.org). Durante l'installazione su Windows, assicurati di selezionare l'opzione "Add Python to PATH".
   - **Git:** Per clonare il repository del progetto. (Puoi scaricarlo da git-scm.com).

2. **Clonazione del Repository:** Apri il tuo terminale (Prompt dei Comandi su Windows, Terminale su macOS/Linux) e segui questi passaggi:
   - Naviga nella directory dove desideri clonare il progetto: *cd C:\Users\USER\Desktop* (Sostituisci il percorso con la tua directory preferita)

- Clona il repository del progetto: *git clone https://github.com/GaeMale/Pianificazione_urbana.git*
- Naviga nella directory del progetto appena clonata: *cd Pianificazione_urbana*

3. **Configurazione dell'Ambiente Virtuale (Raccomandato):** È altamente consigliato utilizzare un ambiente virtuale per isolare le dipendenze del progetto.
    - **Crea l'ambiente virtuale:** Assicurati di essere nella directory principale del progetto (Progetto_pianificazione_urbana) e lancia da terminale il comando `python -m venv venv`

      Questo creerà una sottocartella chiamata venv all'interno della directory

    - **Attiva l'ambiente virtuale:**
        - **Su Windows (Prompt dei Comandi):** `.\venv\Scripts\activate`
        - **Su macOS / Linux (Terminale):** `source venv/bin/activate`

      Dovresti vedere (venv) (o il nome che hai dato al tuo ambiente virtuale) apparire all'inizio del tuo prompt, indicando che l'ambiente è attivo.

4. **Installazione delle Dipendenze:** Una volta attivato l'ambiente virtuale, installa tutte le librerie Python necessarie elencate nel file *requirements.txt*, mediante il seguente comando:

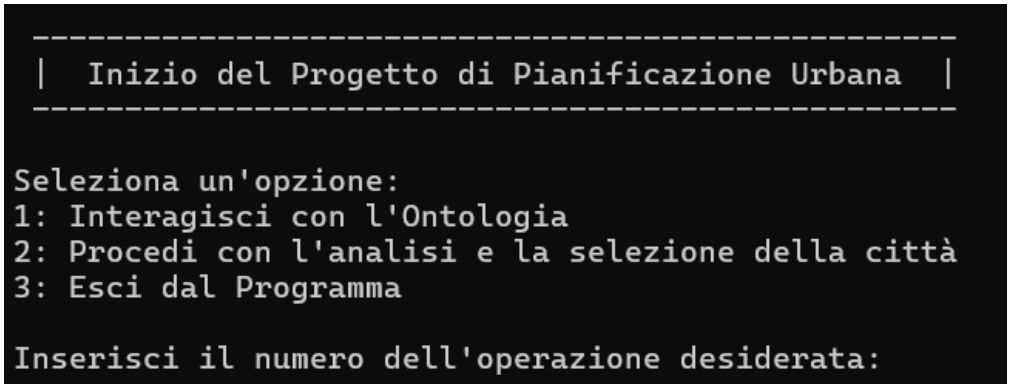
   ```
   pip install -r requirements.txt
   ```

   N.B: Assicurati di essere nella directory principale del progetto

5. Avvio del Progetto: Il progetto può essere avviato eseguendo lo script principale, che è `main.py`.
    - **Avvio tramite Terminale (con ambiente virtuale attivo):**
        - Assicurati che il tuo ambiente virtuale sia attivo (dovresti vedere (venv) nel prompt).
        - Naviga alla directory principale del progetto.
        - Esegui lo script principale: `python main.py`

   Il progetto inizierà a eseguire le sue operazioni, che includeranno il caricamento dei dati, l'analisi, la modellazione e la generazione dei risultati/raccomandazioni.

   Di seguito riportiamo alcune schermate:



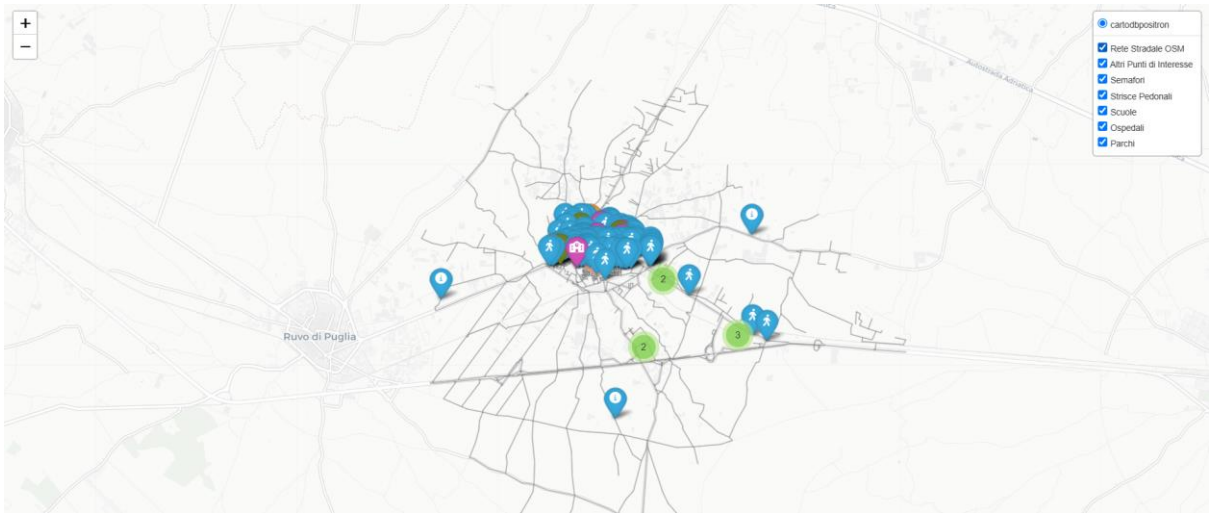
Si richiede all'utente di selezionare un'opzione

```
Inserisci il numero dell'operazione desiderata: 1

Caricamento dell'ontologia...
Ontologia 'ontology/Ontologia_PianificazioneUrbana.rdf' caricata con successo.

Seleziona un'operazione sull'Ontologia:
1: Visualizzazione Classi
2: Visualizzazione Object properties
3: Visualizzazione Data properties
4: Esegui query predefinita
5: Torna al menu principale

Inserisci il numero dell'operazione desiderata:
```

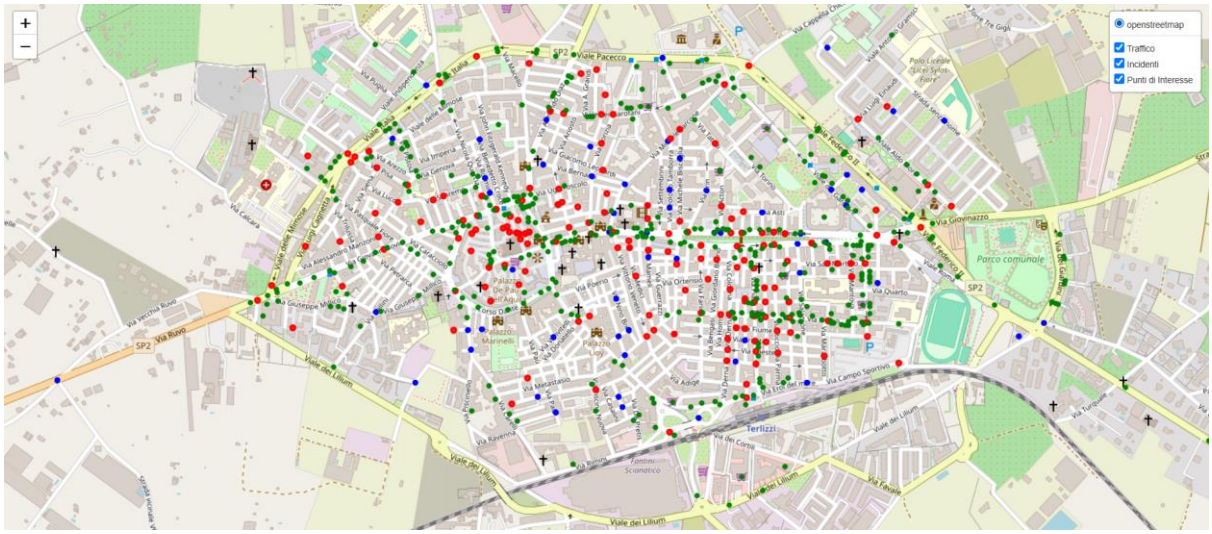
Nel caso in cui abbia digitato '1', si può interagire con l'ontologia

```
Inserisci il numero dell'operazione desiderata: 2

Seleziona la città:
0: Terlizzi, Italy
1: Molfetta, Italy
2: Bari, Italy

Inserisci il numero corrispondente alla città:
```

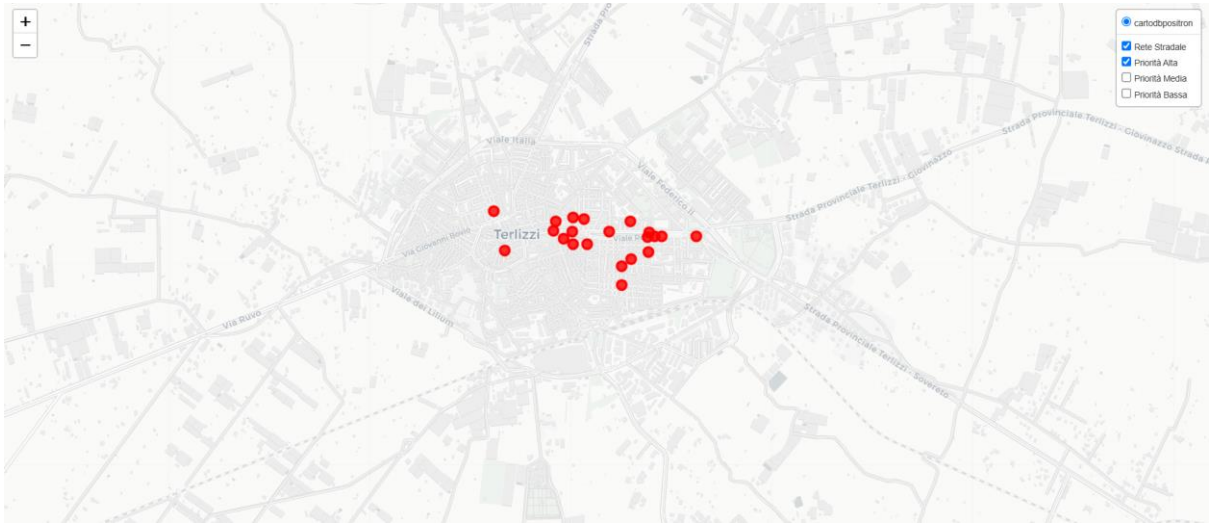
Nel caso in cui abbia digitato '2', si richiede all'utente la città da analizzare (in questo caso sono state considerate unicamente Terlizzi, Molfetta e Bari).



Esempio rete stradale scaricata per la città di Terlizzi



Esempio mappa di distribuzione dati di incidenti (rosso) e traffico (blu). I punti verdi sono i POI (Punti di Interesse).



Esempio mappa generata per le raccomandazioni (di default sarà spuntata solo la voce per i nodi con priorità alta).

N.B: Per funzionare correttamente, il sistema richiede che i dataset di incidenti e traffico siano forniti in un formato specifico nella directory di input (*/data*). In caso di assenza, verranno generati (nel modo più accurato possibile) dataset fittizi per garantire l'esecuzione del modello.

3 Dataset utilizzati e Generazione Dati

Per l'addestramento, la valutazione e il funzionamento del modello di analisi del rischio stradale, il progetto si avvale di diversi tipi di dataset. È fondamentale la disponibilità di dati reali e strutturati, ma il sistema è anche dotato di capacità di generazione di dati fittizi per garantire la sua operatività in contesti dove i dati reali potrebbero essere scarsi o non disponibili.

3.1 Dati della Rete Stradale (OpenStreetMap)

- **Fonte:** Questi dati vengono acquisiti direttamente da **OpenStreetMap (OSM)** tramite la libreria *Osmnx*.
- **Contenuto:** Rappresentano la topologia della rete stradale dell'area urbana specificata (es. una città come Bari o Terlizzi). Includono informazioni su:
    - **Nodi:** Punti di intersezione o capolinea della strada. Ogni nodo ha un osmid (OpenStreetMap ID) unico, coordinate geografiche (latitudine e longitudine). Possono includere attributi come highway (tipo di incrocio), traffic_signals (presenza di semafori), etc.
    - **Edge (Archi):** Segmenti stradali che connettono i nodi. Ogni arco ha un osmid (della via), oneway (senso unico), lanes (numero di corsie), length (lunghezza), highway (tipo di strada), maxspeed (limite di velocità), etc.
- **Ruolo nel Progetto:** Costituisce la base topologica su cui vengono mappati gli incidenti, i flussi di traffico e i POI. Le caratteristiche degli archi e dei nodi (es. numero di corsie, presenza di semafori) sono feature cruciali per il modello predittivo.

```
<?xml version='1.0' encoding='utf-8'?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd"><key id="d21" for="edge" attr.name="access" attr.type="string"/>
<key id="d20" for="edge" attr.name="junction" attr.type="string"/>
<key id="d19" for="edge" attr.name="bridge" attr.type="string"/>
<key id="d18" for="edge" attr.name="tunnel" attr.type="string"/>
<key id="d17" for="edge" attr.name="ref" attr.type="string"/>
<key id="d16" for="edge" attr.name="maxspeed" attr.type="string"/>
<key id="d15" for="edge" attr.name="geometry" attr.type="string"/>
<key id="d14" for="edge" attr.name="lanes" attr.type="string"/>
<key id="d13" for="edge" attr.name="length" attr.type="string"/>
<key id="d12" for="edge" attr.name="reversed" attr.type="string"/>
<key id="d11" for="edge" attr.name="oneway" attr.type="string"/>
<key id="d10" for="edge" attr.name="name" attr.type="string"/>
<key id="d9" for="edge" attr.name="highway" attr.type="string"/>
<key id="d8" for="edge" attr.name="osmid" attr.type="string"/>
<key id="d7" for="node" attr.name="highway" attr.type="string"/>
<key id="d6" for="node" attr.name="street_count" attr.type="string"/>
<key id="d5" for="node" attr.name="x" attr.type="string"/>
<key id="d4" for="node" attr.name="y" attr.type="string"/>
<key id="d3" for="graph" attr.name="simplified" attr.type="string"/>
<key id="d2" for="graph" attr.name="crs" attr.type="string"/>
<key id="d1" for="graph" attr.name="created_with" attr.type="string"/>
<key id="d0" for="graph" attr.name="created_date" attr.type="string"/>
```

Anteprima del file *.graphml* (sarà visualizzato poi sul browser mediante apposita mappa)

3.2 Dati Incidenti Stradali

- **Fonte:** Idealmente, questo dataset proviene da **fonti ufficiali di dati sugli incidenti** (es. database della polizia municipale, ISTAT per l'Italia).
- **Formato Atteso:** Un file strutturato (.xlsx) con una colonna che contenga informazioni sulla gravità dell'incidente e colonne per le coordinate geografiche (latitudine e longitudine) che permettano la geolocalizzazione precisa dell'incidente.
- **Contenuto Tipico:**
    - Latitudine, Longitudine: Coordinate geografiche del luogo dell'incidente.
    - Data, Ora: Data e ora dell'incidente (utile per analisi temporali, anche se non sempre usata direttamente nel modello).

- Gravita: Una metrica della gravità dell'incidente (es. scala numerica da 1 a 4, indicante categorie come "Lievi", "Con feriti", "Mortali"). Nel progetto, viene mappato a un valore numerico per l'aggregazione.
- Causa: Descrizione della causa presunta o accertata dell'incidente (es. "Mancato rispetto precedenza", "Velocità eccessiva", "Distrazione alla guida", ecc.). Questi dati categorici possono essere trasformati in feature one-hot encoded o altre rappresentazioni numeriche per il modello.
- Tipo Veicolo: Identifica i tipi di veicoli coinvolti nell'incidente (es. "Auto", "Bicicletta", “Moto”, ecc.). Anche questa è una variabile categorica utile per comprendere i profili di rischio specifici degli incroci.
- CondizioniMeteo: Descrive le condizioni meteorologiche al momento dell'incidente (es. "Sereno", "Pioggia", "Neve", "Nebbia"). Le condizioni avverse possono aumentare il rischio e la gravità degli incidenti, fornendo un contesto ambientale importante per l'analisi.

- **Ruolo nel Progetto:** Questi dati vengono utilizzati per:
  - Calcolare num_incidenti_vicini: Il conteggio degli incidenti avvenuti in prossimità di ciascun nodo della rete stradale.
  - Calcolare gravita_media_incidente: La gravità media degli incidenti associati a ciascun nodo.
  - Fornire la "verità" per l'addestramento del modello di Machine Learning, indicando quali nodi sono storicamente associati a un rischio.
  - Le colonne aggiuntive (Causa, Tipo Veicolo, CondizioniMeteo) possono essere utilizzate come feature predittive aggiuntive o per analisi descrittive più approfondite.

	A	B	C	D	E	F	G	H	I
1	**Latitudine**	**Longitudine**	**Data**	**Ora**	**Gravita**	**Causa**	**Tipo Veicolo**	**CondizioniMeteo**	**geometry**
2	41,1307739	16,5483921	21/08/2024	09:16	3	mancato_rispetto_stop	camion	Pioggia	POINT (16.5483921 41.130773899999994)
3	41,1310056	16,5390782	08/02/2024	05:20	2	eccesso_velocita	moto	Nebbia	POINT (16.5390782 41.13100559999999)
4	41,1307448	16,538072	16/04/2024	09:28	1	mancanza_precedenza	moto	Pioggia	POINT (16.538072 41.1307448)
5	41,1291437	16,5402679	01/01/2024	15:11	1	mancanza_precedenza	camion	Nebbia	POINT (16.5402679 41.129143700000014)

Prime 5 righe del dataset relativo agli incidenti (**N.B:** La colonna *geometry* sarà costruita dinamicamente a partire dalle colonne Latitudine e Longitudine)

3.3 Dati Traffico

- **Fonte:** Idealmente, questi dati provengono da **stazioni di conteggio traffico**, sensori stradali o modelli di traffico esistenti.
- **Formato Atteso:** Un file strutturato (.xlsx) con una colonna per l'identificatore del nodo/segmento stradale (o coordinate) e colonne per le informazioni sui veicoli.
- **Contenuto Tipico:**
  - IDSensore: Identificatore unico della stazione o del sensore di rilevamento traffico.
  - Latitudine, Longitudine: Coordinate geografiche precise del punto di rilevamento del traffico.
  - DataRilevamento, OraRilevamento: Data e ora specifiche della rilevazione del traffico. Queste colonne consentono analisi temporali (es. medie orarie, giornaliere, settimanali).
  - ConteggioVeicoli: Il volume di veicoli rilevato nel periodo di tempo considerato dal sensore. Questa è la metrica principale utilizzata per valutare l'esposizione al traffico.

    - VelocitaMedia: La velocità media dei veicoli rilevati nel punto e nel periodo specificato. Una velocità media elevata, soprattutto in contesti urbani, può essere un indicatore di rischio.
    - IndiceCongestione: Un valore che indica il livello di congestione del traffico nel punto di rilevamento. Un indice di congestione elevato suggerisce un traffico lento e potenzialmente più caotico, che può influenzare il tipo e la frequenza degli incidenti.
- **Ruolo nel Progetto:** Le metriche di traffico (ConteggioVeicoli, VelocitaMedia, IndiceCongestione) vengono aggregate (es. calcolo della media o del massimo per nodo) e associate ai nodi della rete stradale. L'avg_conteggio_veicoli è una feature cruciale che influenza il rischio di incidente, poiché un maggiore volume di traffico generalmente implica una maggiore esposizione al rischio. VelocitaMedia e IndiceCongestione possono ulteriormente affinare la comprensione del contesto del traffico.

	A	B	C	D	E	F	G	H	I
1	IDSensore	Latitudine	Longitudine	DataRilevamento	OraRilevamento	ConteggioVeicoli	VelocitaMedia	IndiceCongestione	geometry
2	SENSOR_TERLIZZI, ITALY_000	41,1309157	16,5509371	26/02/2024	18:30	62	36,72	0,51	POINT (16.5509371 41.13091569999999)
3	SENSOR_TERLIZZI, ITALY_000	41,1309157	16,5509371	28/08/2024	06:48	94	48,18	0,36	POINT (16.5509371 41.13091569999999)
4	SENSOR_TERLIZZI, ITALY_000	41,1309157	16,5509371	04/01/2024	10:27	196	45,21	0,45	POINT (16.5509371 41.13091569999999)
5	SENSOR_TERLIZZI, ITALY_000	41,1309157	16,5509371	23/11/2024	10:32	154	32,17	0,62	POINT (16.5509371 41.13091569999999)

Prime 5 righe del dataset relativo al traffico (**N.B:** La colonna *geometry* sarà costruita dinamicamente a partire dalle colonne Latitudine e Longitudine)

3.4 Dati Punti di Interesse (POI)

- **Fonte:** Questi dati vengono estratti direttamente da OpenStreetMap tramite interrogazioni specifiche o da database geografici locali.
- **Contenuto Tipico:** Per ogni nodo della rete stradale, vengono calcolati i conteggi di infrastrutture e punti di interesse specifici nelle vicinanze. Le colonne includono:
    - num_pois_vicini: Il conteggio totale di tutti i tipi di Punti di Interesse entro una certa distanza dal nodo. Indica la densità di attività e di potenziale movimento pedonale/veicolare nella zona.
    - num_attraversamenti_pedonali_vicini: Il numero di attraversamenti pedonali presenti entro una definita distanza dal nodo. Un numero elevato può indicare una maggiore interazione tra veicoli e pedoni, che può aumentare il rischio di specifici tipi di incidenti.
    - num_scuole_vicine: Il conteggio delle istituzioni scolastiche (scuole materne, elementari, medie, superiori) nelle immediate vicinanze del nodo. Le zone scolastiche sono spesso associate a orari di punta di traffico pedonale e veicolare, e a utenze vulnerabili (bambini, studenti).
    - num_negozi_vicini: Il numero di attività commerciali al dettaglio (negozi) presenti entro la distanza definita dal nodo. Una concentrazione di negozi può indicare un'area con elevata attività pedonale e veicolare, con frequenti fermate e partenze, e manovre di parcheggio.
- **Ruolo nel Progetto:** Queste feature sono essenziali per arricchire il contesto di ogni nodo, catturando aspetti dell'ambiente urbano che influenzano direttamente la probabilità e la tipologia di incidenti, anche in assenza di dati storici diretti. Contribuiscono in modo significativo al calcolo dell'indice di rischio e alla predizione del modello. Si potrebbero integrare anche altre feature per rendere il modello ancora più sofisticato.

Anteprima del file *.geojson* (sarà combinato con la rete stradale per visualizzare il tutto sul browser)

3.5 Generazione di Dati Fittizi

Il progetto include un modulo per la generazione di dataset sintetici di incidenti e traffico. Questo è fondamentale per:

- **Test e Dimostrazione:** Permette di testare il funzionamento dell'intera pipeline e di dimostrare le capacità del modello anche in assenza di dati reali completi o per aree non coperte da dataset ufficiali.
- **Sviluppo e Debug:** Facilita lo sviluppo e il debug dei moduli, permettendo di controllare le caratteristiche dei dati di input.
- **Pianificazione per scenari:** Consente di simulare diversi scenari di traffico o distribuzione di incidenti per analizzare l'impatto sul rischio.

Questi dataset fittizi vengono generati con distribuzioni e correlazioni che mirano a replicare comportamenti realistici, sebbene non riflettano la precisione e le specificità di un dataset reale.

3.6 Preprocessing del dataset

Nel nostro progetto di Machine Learning per la predizione del rischio stradale, una fase cruciale e imprescindibile è il pre-processing dei dati. Questa fase si occupa di pulire, trasformare e preparare i dati grezzi provenienti da diverse fonti (OpenStreetMap, database incidenti, dati di traffico, POI, ecc.) in un formato che sia comprensibile e utilizzabile dal nostro modello di Machine Learning.

Per garantire la coerenza, la riproducibilità e la modularità di questo processo, abbiamo incapsulato tutte le operazioni di pre-processing all'interno di una classe Python dedicata, denominata *preprocessing.py*.

3.6.1 Features presenti nel dataset

Il nostro modello di Machine Learning per l'identificazione dei nodi a rischio sulla rete stradale si basa su un set mirato di feature di input. Queste caratteristiche sono state selezionate per la loro capacità di descrivere in modo efficace il contesto infrastrutturale, di traffico e urbano di ogni nodo, fattori che influenzano direttamente la probabilità di incidenti.

L'obiettivo è catturare la complessità di un nodo stradale con un set di variabili che siano sia informative che gestibili.

- **Feature Utilizzate nel Modello:**
    1. **grado_incrocio:**
        - Descrizione: Questa feature numerica rappresenta il "grado" di complessità di un incrocio, tipicamente il numero di rami (o strade) che convergono in quel nodo. Un incrocio con più strade che si intersecano è intrinsecamente

più complesso da gestire per i conducenti e può aumentare i punti di conflitto.
    - Rilevanza: Un grado più alto suggerisce una maggiore complessità strutturale e potenziale per manovre pericolose, interazioni veicolari e pedonali complesse, e quindi un rischio maggiore.
2. **num_pois_vicini:**
    - Descrizione: Un conteggio del numero di "Punti di Interesse" (Points of Interest) generici, come negozi, parchi o altre attività, presenti in un raggio predefinito attorno al nodo stradale.
    - Rilevanza: Un'alta concentrazione di POI indica un'area ad alta attività urbana, che si traduce in un aumento del traffico veicolare (clienti, consegne) e pedonale, incrementando le opportunità di conflitto e incidenti.
3. **num_attraversamenti_pedonali_vicini:**
    - Descrizione: Il numero di attraversamenti pedonali (strisce pedonali) situati nelle immediate vicinanze del nodo.
    - Rilevanza: La presenza e la densità di attraversamenti pedonali è direttamente correlata al flusso di pedoni. Maggiori attraversamenti significano più interazioni tra veicoli e pedoni, aumentando il rischio di incidenti che coinvolgono utenti vulnerabili della strada.
4. **num_scuole_vicine:**
    - Descrizione: Il conteggio delle scuole o istituti educativi presenti nel raggio di prossimità del nodo.
    - Rilevanza: Le aree vicino alle scuole sperimentano picchi di traffico (veicolare e pedonale) in specifici orari del giorno (ingresso/uscita). La presenza di bambini e adolescenti aumenta la vulnerabilità e richiede maggiore attenzione da parte dei conducenti, influenzando il profilo di rischio.
5. **num_negozi_vicini:**
    - Descrizione: Il numero di attività commerciali (negozi, supermercati, centri commerciali) situati nell'area circostante il nodo.
    - Rilevanza: Simile ai POI, un'alta densità di negozi genera traffico di clienti, veicoli per consegne, e spesso manovre di parcheggio/fermata, tutti fattori che possono contribuire all'aumento del rischio.
6. **avg_conteggio_veicoli:**
    - Descrizione: La media del conteggio dei veicoli che transitano attraverso il nodo in un determinato periodo (es. media giornaliera o oraria di punta).
    - Rilevanza: Un volume di traffico più elevato aumenta statisticamente il numero di interazioni tra veicoli e, di conseguenza, la probabilità di collisioni.
7. **avg_velocita:**
    - Descrizione: La velocità media dei veicoli che attraversano il nodo.
    - Rilevanza: La velocità è un fattore critico nella gravità degli incidenti. Velocità medie elevate possono indicare un rischio maggiore di incidenti gravi. Al contrario, velocità molto basse potrebbero indicare congestione che, pur riducendo la gravità, può aumentare la frequenza di tamponamenti.
8. **avg_indice_congestione:**
    - Descrizione: Un valore che quantifica il livello medio di congestione del traffico nel nodo. Può essere basato su ritardi, rapporto volume/capacità, o altre metriche di fluidità.
    - Rilevanza: Un'alta congestione può portare a comportamenti di guida frustrati, cambi di corsia frequenti, frenate brusche e aumento delle

interazioni ravvicinate, tutti fattori che contribuiscono al rischio di incidenti, in particolare quelli a bassa velocità o tamponamenti.

- **Feature Rimosse e Motivazioni:** Durante la fase di preparazione del dataset e feature engineering, abbiamo deciso di escludere alcune feature inizialmente considerate. Questa scelta è stata guidata dalla necessità di ottimizzare le prestazioni del modello e di garantire la coerenza logica dei dati:
    1. **osmid:**
        - Motivazione: L'ID di OpenStreetMap (osmid) è un identificatore univoco per ciascun nodo. Sebbene sia utile per identificare e localizzare i nodi, non porta alcuna informazione predittiva diretta per il modello. È un semplice identificativo e la sua inclusione sarebbe un rumore inutile o, peggio, potrebbe portare il modello a "memorizzare" ID specifici invece di apprendere pattern generalizzabili. Viene mantenuto come riferimento esterno per la mappatura, ma non come feature di input.
    2. **num_incidenti_vicini** e **ha_incidente_vicino:**
        - Motivazione: Queste feature rappresentano il rischio storico o passato di un nodo (il numero di incidenti avvenuti in precedenza e se un incidente è avvenuto o meno). Nel nostro problema di predizione, queste sono le variabili che il nostro modello deve imparare a prevedere (la nostra y). Includerle direttamente come feature di input sarebbe una forma di ***data leakage*** (o "fuoriuscita di dati"). Il modello imparerebbe semplicemente a copiare questa informazione, invece di imparare a prevedere il rischio basandosi su altre caratteristiche del nodo. Questo porterebbe a prestazioni artificialmente elevate sul training set, ma il modello sarebbe inutile per predire il rischio di nodi nuovi per cui non si hanno dati storici di incidenti.

Questa attenta selezione delle feature e la motivata esclusione di altre ci hanno permesso di costruire un modello che non solo è predittivo, ma anche interpretabile e robusto, focalizzato sull'identificazione del rischio basato su caratteristiche oggettive e generalizzabili dei nodi stradali.

3.6.2 Feature Engineering

Nel nostro progetto di predizione del rischio stradale, il **Feature Engineering** è stato un processo fondamentale per estrarre informazioni più significative e predittive dai dati grezzi iniziali. L'obiettivo era trasformare le informazioni disponibili su nodi stradali, incidenti e contesto urbano in feature numeriche e categoriche che i modelli di Machine Learning potessero efficacemente utilizzare per identificare i pattern di rischio.

Come è stato Implementato il Feature Engineering: L'implementazione del Feature Engineering è avvenuta principalmente attraverso la creazione di nuove variabili a partire da quelle esistenti, o tramite l'aggregazione di informazioni spaziali e temporali:

- **Aggregazione Spaziale e Conteggio di Prossimità:**
    - num_pois_vicini: Questa feature non è un dato grezzo, ma è stata creata contando il numero di Punti di Interesse (POI) entro un certo raggio (es. 100-200 metri) da ogni nodo stradale. Ha richiesto l'uso di dati geografici e calcoli di vicinanza.

- num_attraversamenti_pedonali_vicini: Similmente, questa feature è il risultato del conteggio degli attraversamenti pedonali rilevati nelle vicinanze di ciascun nodo, estratto da dati OpenStreetMap.
- num_scuole_vicine: Conteggio delle scuole in un raggio specifico.
- num_negozi_vicini: Conteggio dei negozi in un raggio specifico.

Queste feature trasformano la presenza "sparse" di POI o infrastrutture in una densità quantificabile, rendendola una metrica predittiva per il modello, mediante utilizzo di librerie (come geopandas, shapely) per calcoli di prossimità, buffer e intersezioni.

- **Normalizzazione o Aggregazione di Dati di Traffico:**
  - avg_conteggio_veicoli: Implica l'aggregazione di dati di conteggio veicoli su un periodo (es. giornaliero, orario di punta) per ottenere una media, rendendo la feature più stabile e rappresentativa del flusso tipico.
  - avg_velocita: Similmente, la velocità media è una metrica aggregata che riassume il comportamento tipico del traffico in quel nodo.
  - avg_indice_congestione: Questo indice è di per sé una feature "ingegnerizzata", spesso calcolata a partire da velocità, volumi e capacità stradale.

- **Misura di "Connettività" Stradale:**
  - grado_incrocio: Questa feature è stata derivata direttamente dalla topologia del grafo stradale (da OpenStreetMap), contando il numero di "edge" (segmenti stradali) collegati a un "node" (incrocio). Trasforma una relazione topologica in una metrica numerica utile.

3.7 Bilanciamento delle classi

Mediante un'istruzione di stampa, viene mostrata a schermo la distribuzione della variabile target ***ha_incidente_vicino*** (es. per un run con dati fittizi per la città di Bari):

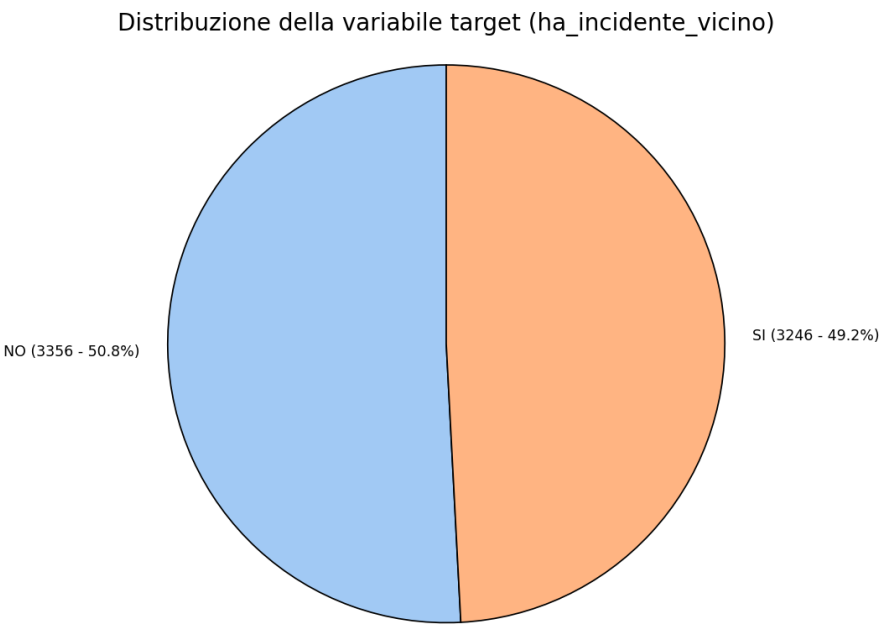
```
Distribuzione ha_incidente_vicino:
ha_incidente_vicino
0    3356
1    3246
```

Questo output testuale illustra la distribuzione dei conteggi delle osservazioni per ciascuna delle due categorie all'interno della variabile *ha_incidente_vicino*. Questa variabile riveste un ruolo cruciale, in quanto costituisce la variabile target che il modello di Machine Learning è addestrato a prevedere.

- Il valore 0 nella colonna *ha_incidente_vicino* indica la classe "negativa". In questo contesto, rappresenta i nodi stradali che non hanno registrato incidenti nelle loro vicinanze entro il periodo di riferimento, o che non sono stati classificati come "a rischio". Il conteggio di 3356 osservazioni per questa classe indica la sua frequenza nel dataset.
- Il valore 1 nella stessa colonna indica la classe "positiva". Questo valore designa i nodi stradali che hanno registrato incidenti nelle loro vicinanze e sono quindi classificati come "a rischio" o "hotspot". Per questa classe, sono presenti 3246 osservazioni.

L'analisi di questa distribuzione rivela un aspetto fondamentale del dataset: le due classi della variabile target *ha_incidente_vicino* sono sostanzialmente bilanciate (in questo caso). Con 3356 casi di "non rischio" e 3246 casi di "rischio", la differenza tra il numero di osservazioni per la classe positiva e quella negativa è minima. Questa condizione di equilibrio è particolarmente favorevole per l'addestramento dei modelli di Machine Learning. Un dataset bilanciato tende a prevenire che il modello sviluppi un bias verso la classe maggioritaria, un problema comune nei dataset sbilanciati che spesso richiede tecniche di pre-processing aggiuntive (come l'*oversampling* o l'*undersampling*) per essere mitigato. La distribuzione quasi paritaria assicura che il modello possa apprendere in modo efficace le caratteristiche distintive di entrambe le classi, contribuendo a una maggiore robustezza e affidabilità delle previsioni.

Di seguito la stessa distribuzione in forma “grafica”:



4 Apprendimento Supervisionato

Nel nostro progetto di analisi e pianificazione della sicurezza stradale, l'**apprendimento supervisionato** è il paradigma di Machine Learning utilizzato per addestrare il modello a identificare e prevedere i nodi della rete stradale con un alto potenziale di rischio incidente.

4.1 Divisione Training set e Test set

Nel nostro progetto di Machine Learning per la predizione del rischio stradale, una fase importante è stata la suddivisione del dataset completo in **Training Set** e **Test Set**. Questa operazione è fondamentale per garantire che la valutazione delle prestazioni del nostro modello sia robusta e imparziale, riflettendo accuratamente la sua capacità di operare su dati reali e non precedentemente incontrati.

L'obiettivo principale della suddivisione è evitare l'**overfitting**. Se un modello venisse addestrato e poi valutato sullo stesso identico set di dati, le sue prestazioni apparirebbero artificialmente elevate. Il modello avrebbe semplicemente "memorizzato" le risposte, piuttosto che "imparare" i pattern sottostanti che gli consentirebbero di fare previsioni accurate su nuove osservazioni.

- **Training Set:** Questo sottoinsieme di dati viene utilizzato esclusivamente per addestrare il modello. È la "scuola" del modello, dove impara le relazioni tra le feature di input (le caratteristiche del nodo stradale) e la variabile target (ha_incidente_vicino - se il nodo è a rischio o meno).
- **Test Set:** Questo sottoinsieme è tenuto completamente separato e non visto dal modello durante l'addestramento. Funge da "esame finale" del modello. Le metriche di performance calcolate su questo set forniscono una stima realistica di come il modello si comporterà una volta messo in produzione, su dati nuovi e sconosciuti.

Dettagli dell'Implementazione nel Nostro Caso: La suddivisione è stata implementata utilizzando la funzione *train_test_split* dalla libreria *sklearn.model_selection* di Python, con parametri specifici scelti per adattarsi alle caratteristiche del nostro dataset e agli obiettivi del progetto:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

- **X e y:**
  - X rappresenta il nostro DataFrame di feature di input, contenente tutte le caratteristiche del nodo stradale (es. grado_incrocio, num_pois_vicini, avg_conteggio_veicoli, ecc.).
  - y è la nostra variabile target
- **test_size=0.2:**
  - Questo parametro specifica che il 20% del dataset totale (sia X che y) è stato assegnato al Test Set. Di conseguenza, il restante 80% è stato utilizzato per il Training Set. Questa proporzione è una scelta comune che bilancia la necessità di avere sufficienti dati per addestrare il modello e quella di avere un test set abbastanza grande per una valutazione affidabile.
- **random_state=42:**
  - Il random_state è un parametro per il generatore di numeri pseudo-casuali utilizzato per la divisione. Impostarlo su un valore fisso (come 42) è cruciale per la riproducibilità del nostro lavoro. Significa che ogni volta che il codice viene eseguito, la divisione del dataset sarà esattamente la stessa. Questo elimina la variabilità casuale nelle valutazioni del modello e facilita il debugging e il confronto tra diverse configurazioni del modello.
- **stratify=y:**
  - Questo è un parametro di importanza critica per il nostro problema di classificazione. Indica che la divisione deve essere stratificata in base alla variabile y. Dato che la nostra variabile target ha_incidente_vicino è binaria (0 o 1), la stratificazione assicura che la proporzione delle classi presente nel dataset originale (y) venga mantenuta fedelmente sia nel y_train che nel y_test.
  - Anche se la nostra analisi (sulla città di Bari) ha mostrato che le classi '0' e '1' sono già quasi perfettamente bilanciate nel dataset completo (3356 vs 3246 osservazioni), l'uso di stratify=y fornisce una garanzia aggiuntiva. Previene scenari improbabili (ma possibili senza stratificazione) in cui una classe, anche se ben rappresentata globalmente, potrebbe essere sovra o sotto-rappresentata in uno dei subset a causa di una divisione puramente casuale. Questo assicura che sia il training che il test set siano rappresentativi della reale distribuzione del rischio.

4.2 Scelta del modello

Nel nostro progetto, stiamo confrontando le prestazioni di diversi algoritmi di Machine Learning per identificare il modello più efficace nella predizione del rischio di incidenti stradali. Ogni modello adotta un approccio distinto all'apprendimento dai dati:

- **Random Forest Classifier:** È un algoritmo basato su "ensemble learning" che costruisce un gran numero di alberi decisionali durante l'addestramento e restituisce la classe che è la moda delle classi (la più frequente) dei singoli alberi. È robusto all'overfitting e gestisce bene le relazioni non lineari.

- **Logistic Regression:** Modella la probabilità che un'istanza appartenga a una determinata classe utilizzando una funzione logistica. È semplice, veloce da addestrare e altamente interpretabile.

- **Gradient Boosting Classifier:** Un altro metodo ensemble che costruisce alberi decisionali in sequenza. Ogni nuovo albero cerca di correggere gli errori commessi dall'albero precedente. È molto potente e spesso produce risultati di alta qualità.

- **K-Nearest Neighbors (KNN):** È un algoritmo "lazy learning" non parametrico. Classifica un nuovo punto dati assegnandolo alla classe più comune tra i suoi k vicini più prossimi nel training set. La classificazione si basa sulla distanza.

- **Support Vector Machine (SVM):** Cerca di trovare l'iperpiano ottimale che separi le classi nello spazio delle feature, massimizzando il margine tra i punti dati delle diverse classi.

Dato il tipo di feature utilizzate nel progetto (molte delle quali sono intrinsecamente correlate e dipendenti tra loro nel contesto urbano e stradale), l'algoritmo **Naive Bayes**, con la sua assunzione di indipendenza, sarebbe probabilmente meno adatto a catturare la piena complessità delle relazioni sottostanti rispetto a un modello come il Random Forest.

Per garantire che la stima delle prestazioni del nostro modello sia robusta e affidabile, adottiamo la tecnica del **K-Fold Cross-Validation**, in particolare la sua variante **Stratified K-Fold Cross-Validation**.

- **Cos'è:** Invece di dividere il dataset una sola volta in training e test set, il K-Fold divide l'intero dataset in K sottoinsiemi (chiamati "fold") di dimensioni approssimativamente uguali. Il processo si ripete K volte: in ogni iterazione, un fold diverso viene usato come set di test, mentre i restanti K-1 fold vengono usati come set di addestramento.

- **Stratified K-Fold:** Per il nostro problema, che probabilmente presenta classi sbilanciate (pochi hotspot di rischio rispetto a molti nodi sicuri), la "stratificazione" è cruciale. Assicura che la proporzione delle classi (es. 10% nodi a rischio, 90% non a rischio) sia mantenuta in ogni singolo fold, sia di addestramento che di test. Questo impedisce che un fold di test non contenga esempi della classe minoritaria, rendendo la valutazione inaffidabile.

- **Stima Più Affidabile:** Le metriche di performance (Accuracy, Precision, Recall, F1-Score, AUC-ROC, Average Precision) vengono calcolate per ciascuno dei K cicli e poi mediate. Questo fornisce una stima molto più stabile e meno dipendente dalla specifica divisione casuale dei dati.

- **Migliore Utilizzo dei Dati:** Ogni punto dati nel nostro dataset viene utilizzato esattamente una volta nel set di test e K-1 volte nel set di addestramento, massimizzando l'utilizzo delle informazioni disponibili.

- **Riduzione dell'Overfitting**: Valutando il modello su diversi sottoinsiemi di dati, otteniamo una misura più accurata di quanto bene il modello generalizzerà a dati reali e non visti, riducendo il rischio di overfitting sul set di training originale.

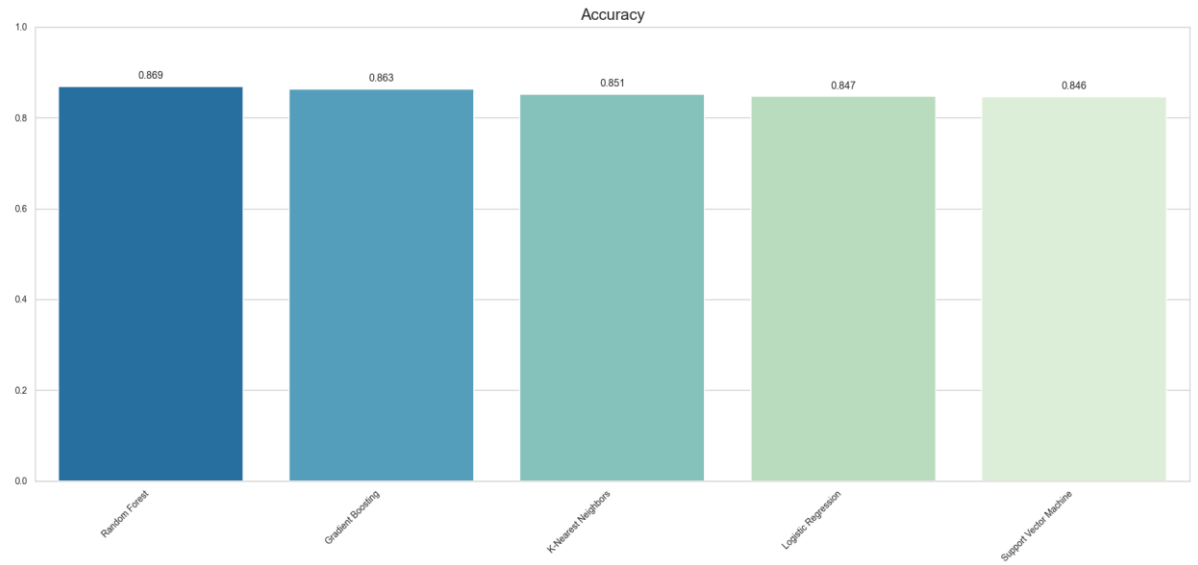
Inoltre, le metriche utilizzate per la valutazione dei vari modelli sono:

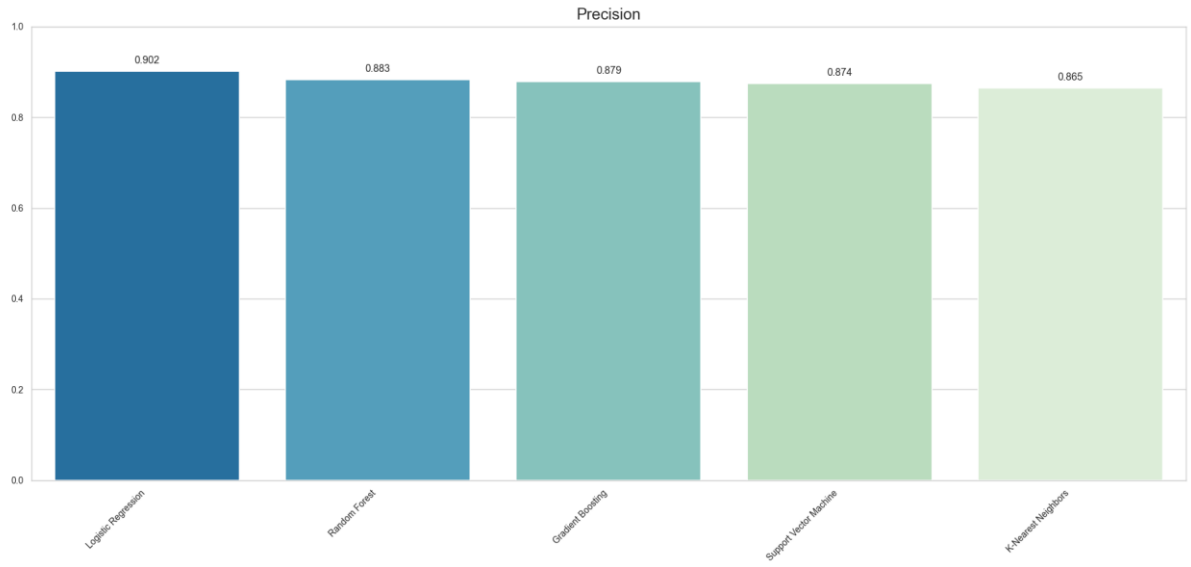
- **Accuratezza:** La percentuale di tutte le previsioni corrette. Fornisce una visione generale di quanto spesso il modello è corretto.
- **Precisione:** Tra tutte le volte che il modello ha predetto la **classe positiva** (es. "rischio"), quante volte aveva ragione.
- **Richiamo:** Tra tutti i casi che *erano realmente* della **classe positiva** (es. i veri "rischio"), quanti il modello è riuscito a identificare correttamente.
- **F1-score:** È la **media armonica** di Precisione e Richiamo. Cerca di fornire un equilibrio tra le due metriche.
- **Average Precision:** È una metrica di valutazione utilizzata per i problemi di **classificazione binaria**, ed è particolarmente rilevante e utile quando si ha a che fare con **dataset sbilanciati**. L'Average Precision si calcola come l'**area sotto la Curva Precision-Recall (PR Curve)**.

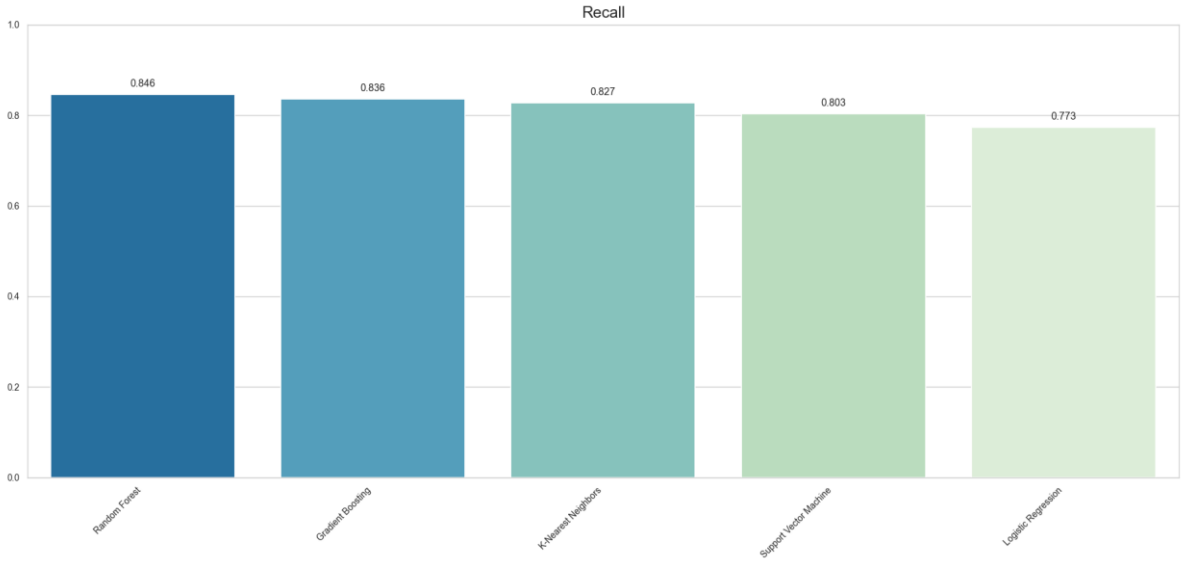
Di seguito è riportata la tabella riassuntiva delle prestazioni medie per tutti i modelli valutati, ottenuta tramite K-Fold Cross-Validation (prendendo come esempio la città di Bari):

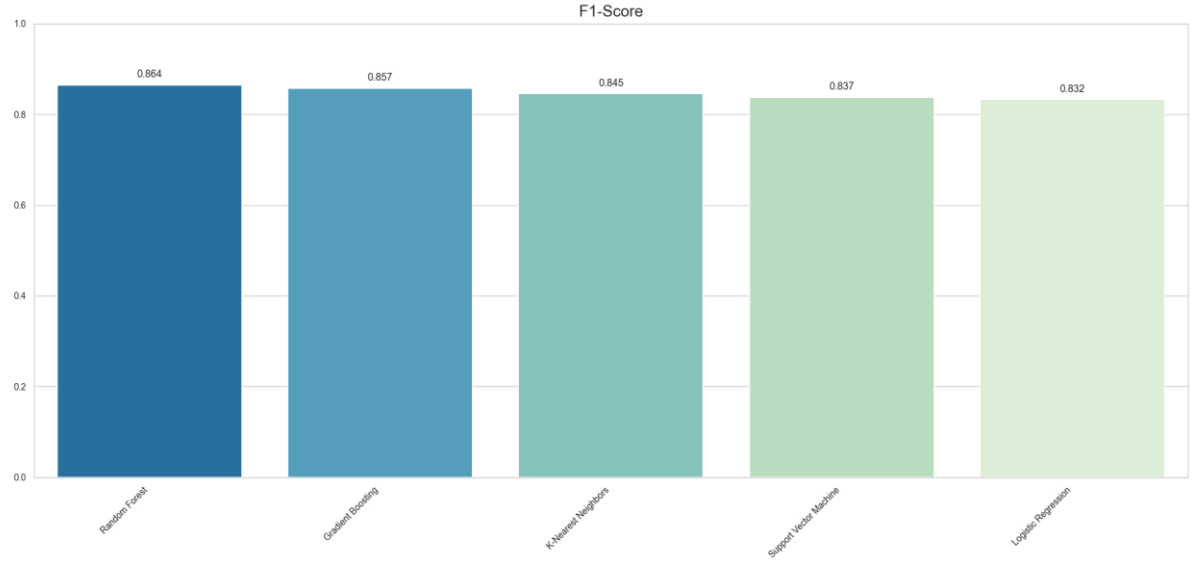
```
--- Confronto delle prestazioni medie dei modelli (K-Fold CV) ---

                Model  Avg Accuracy  Avg Precision  Avg Recall  Avg F1-Score  Avg Average Precision (AP)
        Random Forest      0.869130       0.883231    0.845656      0.864014                    0.938890
  Logistic Regression      0.846864       0.901632    0.772956      0.832279                    0.930075
    Gradient Boosting      0.862617       0.878529    0.836414      0.856896                    0.937222
  K-Nearest Neighbors      0.851257       0.864671    0.827167      0.845407                    0.894280
Support Vector Machine      0.846411       0.874261    0.803147      0.837144                    0.913162
```

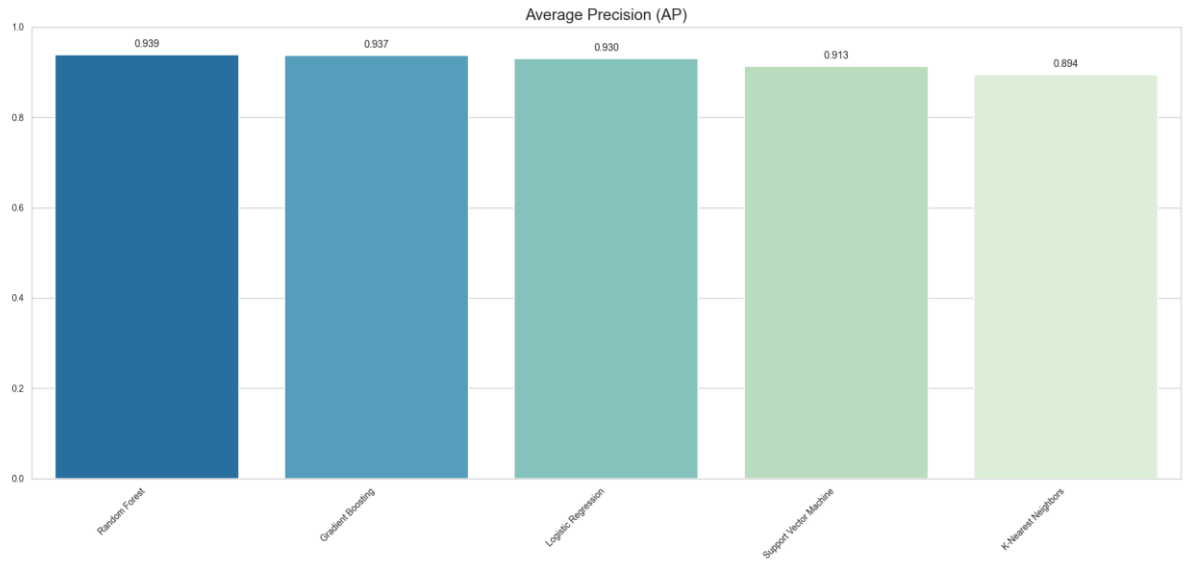
In particolare:




Precision
1.0
0.8
0.6
0.4
0.2
0.0
0.902
0.883
0.879
0.874
0.865
Logistic Regression
Random Forest
Gradient Boosting
Support Vector Machine
K-Nearest Neighbors


Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.846
0.836
0.827
0.803
0.773
Random Forest
Gradient Boosting
K-Nearest Neighbors
Support Vector Machine
Logistic Regression


F1-Score
1.0
0.8
0.6
0.4
0.2
0.0
0.864
0.857
0.845
0.837
0.832
Random Forest
Gradient Boosting
K-Nearest Neighbors
Support Vector Machine
Logistic Regression



Le tabelle confermano che il **Random Forest Classifier** è il modello che ha performato meglio in media su tutte le metriche importanti del progetto. Non solo ha la più alta accuratezza complessiva, ma eccelle anche nel Richiamo, il che è cruciale per bilanciare l'identificazione degli hotspot reali con la riduzione dei falsi allarmi. La sua Avg Average Precision (AP) più alta indica anche che è il più efficace nell'identificare correttamente i nodi a rischio anche in un contesto di dataset sbilanciato.

4.3 Scelta degli iper-parametri

Le prestazioni e il comportamento di un modello Random Forest sono fortemente influenzati dai suoi **iper-parametri**, che sono impostazioni configurate prima della fase di addestramento. Nel nostro caso, abbiamo ottimizzato il RandomForestClassifier utilizzando i seguenti iper-parametri:

- **n_estimators=100**
  - **Descrizione:** Questo iper-parametro definisce il numero di alberi decisionali che verranno costruiti nel Random Forest. Un numero maggiore di alberi tende a migliorare le prestazioni del modello e a renderlo più robusto, poiché la combinazione di più previsioni individuali riduce la varianza.
  - **Giustificazione della Scelta:** Abbiamo scelto 100 alberi come un buon compromesso. Un numero troppo basso potrebbe non sfruttare appieno il potenziale dell'ensemble, mentre un numero eccessivamente elevato (es. 500, 1000) aumenterebbe il tempo di addestramento e inferenza senza necessariamente portare a miglioramenti significativi delle prestazioni dopo un certo punto. 100 è un valore comunemente usato come punto di partenza efficace.
- **random_state=42**
  - **Descrizione:** Questo parametro serve per far sì che il nostro modello di Machine Learning (il Random Forest) lavori sempre allo stesso modo ogni volta che lo addestriamo.
  - **Giustificazione della Scelta:** Impostando random_state=42 (o qualsiasi altro numero intero), ci assicuriamo che i nostri esperimenti siano riproducibili. Questo significa che chiunque esegua il nostro codice con gli stessi dati e gli stessi parametri, otterrà esattamente gli stessi risultati. È fondamentale per la validazione del modello e per apportare miglioramenti in modo coerente.

- **n_jobs=-1**
    - **Descrizione:** Questo parametro specifica il numero di CPU (o core) da utilizzare per l'addestramento del modello. -1 significa che il modello utilizzerà tutti i core disponibili sulla macchina.
    - **Giustificazione della Scelta:** Il Random Forest è un algoritmo che può essere facilmente parallelizzato, poiché la costruzione di ciascun albero decisionale è indipendente dalle altre. L'utilizzo di n_jobs=-1 consente di accelerare significativamente i tempi di addestramento, sfruttando appieno la potenza di calcolo disponibile.
- **class_weight='balanced'**
    - **Descrizione:** Questo iper-parametro è fondamentale per gestire il bilanciamento delle classi nel dataset di addestramento. Quando impostato su 'balanced', l'algoritmo assegna automaticamente dei pesi alle classi in base alla loro frequenza inversa. Le classi meno frequenti (minoritarie) riceveranno un peso maggiore, e quelle più frequenti (maggioritarie) un peso minore.
    - **Giustificazione della Scelta:** Anche se nel nostro caso le classi 0 e 1 sono risultate quasi perfettamente bilanciate (3356 vs 3246 osservazioni), l'inclusione di class_weight='balanced' funge da ulteriore misura di robustezza. In scenari dove un leggero sbilanciamento potrebbe emergere o se il dataset dovesse evolvere diventando più sbilanciato, questa impostazione aiuta il modello a prestare uguale attenzione a entrambe le classi, prevenendo che si focalizzi troppo sulla classe maggioritaria a discapito della minoritaria.
- **max_depth=7**
    - **Descrizione:** Questo iper-parametro controlla la profondità massima di ogni singolo albero decisionale nel Random Forest. Limitarla significa che gli alberi non potranno crescere indefinitamente.
    - **Giustificazione della Scelta:** Un albero troppo profondo tende a fare overfitting sui dati di addestramento, imparando a memoria il rumore invece che i veri pattern generalizzabili. Limitare la max_depth a 7 aiuta a:
        - **Ridurre l'overfitting:** Impedisce agli alberi di diventare troppo complessi e specifici per i dati di addestramento.
        - **Migliorare la generalizzazione:** Promuove l'apprendimento di regole più semplici e ampie che si applicano meglio a dati non visti.
- **min_samples_leaf=5**
    - **Descrizione:** Questo iper-parametro specifica il numero minimo di campioni richiesti per essere in un nodo foglia (leaf node) di un albero. Un nodo foglia è un nodo terminale che non si divide ulteriormente.
    - **Giustificazione della Scelta:** Similmente a max_depth, min_samples_leaf è un meccanismo per controllare la complessità dell'albero e prevenire l'overfitting. Richiedendo un minimo di 5 campioni in ogni foglia, si evitano divisioni che portano a nodi con pochissimi campioni (che potrebbero rappresentare rumore). Questo contribuisce a:
        - **Aumentare la robustezza:** Le decisioni prese nei nodi foglia sono basate su un numero sufficiente di campioni.
        - **Ridurre l'overfitting:** Impedisce agli alberi di creare rami troppo specifici per piccoli sottoinsiemi di dati.

In sintesi, la combinazione di questi iper-parametri è stata attentamente scelta per bilanciare l'accuratezza del modello, la sua robustezza contro l'overfitting, l'efficienza computazionale e la capacità di gestire in modo equo le classi target, rendendo il RandomForestClassifier un'ottima scelta per la predizione del rischio stradale nel nostro contesto.

4.4 Coefficiente di Correlazione di Pearson

Il Coefficiente di Correlazione di Pearson è una misura statistica che quantifica la forza e la direzione della relazione lineare tra due variabili quantitative. Il suo valore varia tra -1 e +1.

Nel nostro caso è stato calcolato tra 'Probabilità di Rischio Predetta' e 'Indice di Rischio', avendo (es. per un run su Bari) un valore di 0.8928:

```
Coefficiente di Correlazione di Pearson tra 'Probabilità di Rischio Predetta' e 'Indice di Rischio': 0.8928
```

- **Forte Relazione Lineare Positiva:** C'è una relazione lineare molto forte e positiva tra la probabilità di rischio che il modello di ML ha predetto e l'indice di rischio misurato.
- **Direzione della Relazione:** All'aumentare della probabilità di rischio predetta dal modello, tende ad aumentare anche l'indice di rischio effettivo, e viceversa. Questo indica che il modello riesce a stimare bene l'indice di rischio, poiché le sue previsioni si muovono nella stessa direzione.

È importante sottolineare che la correlazione (anche se forte) non implica causalità. Tuttavia, in questo caso specifico, l'obiettivo è proprio che la previsione del modello sia correlata al valore reale, quindi una correlazione alta è un segno di bontà del modello predittivo.

Il grafico di seguito mostra come i valori di probabilità_rischio_predetta (asse X) si relazionano con i valori di indice_rischio (asse Y) per ogni singolo nodo. La colorazione progressiva basata sull'indice_rischio è un modo per visualizzare la densità e la distribuzione dei punti.

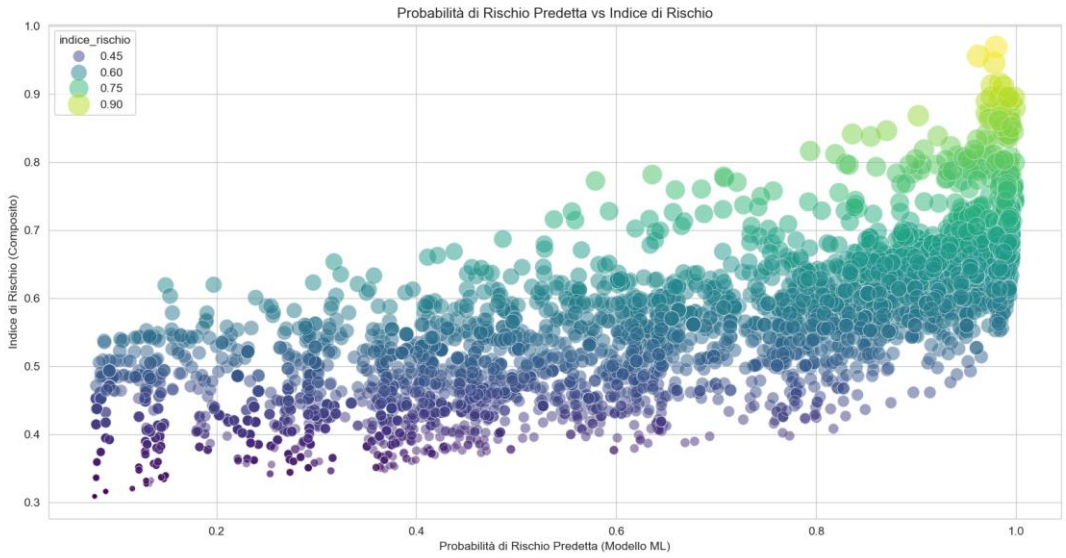


Grafico a dispersione: probabilità predetta vs indice di rischio

5 Ontologia

Un'ontologia, nel contesto dell'informatica e dell'intelligenza artificiale, è una rappresentazione formale ed esplicita di un dominio di conoscenza. È un modello strutturato che definisce:

- **Classi** (o **Concetti**): Le categorie principali di cose o idee nel dominio (es. "Persona", "Veicolo", "Evento").
- **Proprietà** (o **Relazioni**):
  - **Object Properties:** Descrivono le relazioni tra le classi o gli individui (es. "Persona possiede Veicolo").
  - **Data Properties:** Descrivono le relazioni tra le classi/individui e i valori dei dati (es. "Persona ha età", "Veicolo ha colore").
- **Individui** (o **Istanze**): Esempi concreti delle classi (es. "Mario" è un'istanza di "Persona", "Fiat Punto" è un'istanza di "Veicolo").

L'obiettivo principale di un'ontologia è consentire a computer e esseri umani di condividere e comprendere il significato delle informazioni in un dominio specifico. Rendendo la conoscenza esplicita e leggibile dalle macchine, le ontologie facilitano:

- **Interoperabilità:** Sistemi diversi possono scambiare dati sapendo esattamente cosa significano.
- **Condivisione della Conoscenza:** Una base di conoscenza comune per lo sviluppo di applicazioni.
- **Ragionamento Automatico:** I "ragionatori" (software specifici) possono inferire nuove conoscenze da quelle esistenti (es. se "Mario è genitore di Luigi" e "Luigi è genitore di Pietro", un ragionatore può inferire "Mario è nonno di Pietro" se la proprietà "genitore di" è definita come transitiva in un certo modo).
- **Ricerca Semantica:** Query più intelligenti che vanno oltre la semplice corrispondenza di parole chiave.

Le ontologie sono spesso modellate utilizzando linguaggi standard come **OWL** (Web Ontology Language), basato su **RDF** (Resource Description Framework) per la rappresentazione dei dati sul web.

5.1 Protégé

Per la creazione di Ontologia, abbiamo utilizzato **Protégé**, lo strumento standard e più popolare per la creazione e la modifica di ontologie, specialmente quelle basate su OWL e RDF.

È un editor grafico che permette di definire classi, sottoclassi, proprietà (oggetti e tipi di dati) e individui in modo visuale e intuitivo.

Include integrazioni con ragionatori (come HermiT) che possono verificare la logica dell'ontologia e inferire nuove conoscenze. Inoltre, è open-source e ampiamente supportato dalla comunità.

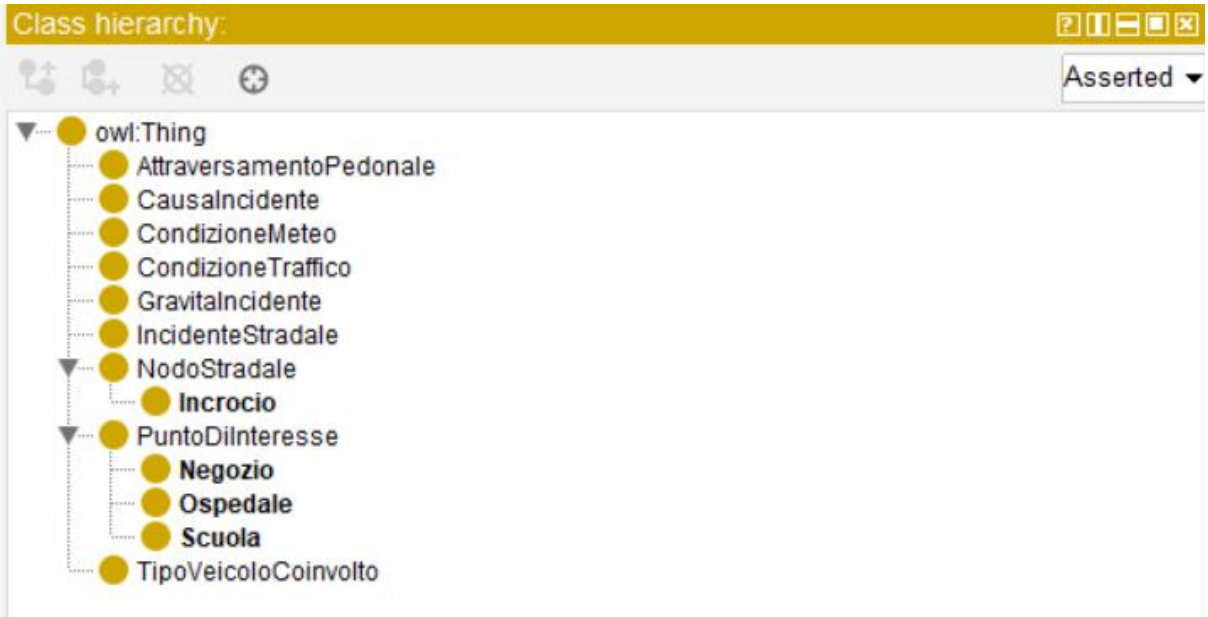
5.2 La nostra ontologia

La nostra ontologia, definita dall'IRI *http://www.semanticweb.org/gaetano/ontologies/2025/6/pianificazioneUrbana*, è un modello formale per rappresentare e analizzare dati relativi agli incidenti stradali e ai contesti urbani ad essi

correlati. Il suo scopo principale è fornire una base di conoscenza strutturata che possa essere utilizzata per la pianificazione urbana, l'analisi della sicurezza stradale e la prevenzione degli incidenti.

Componenti Chiave della Nostra Ontologia:

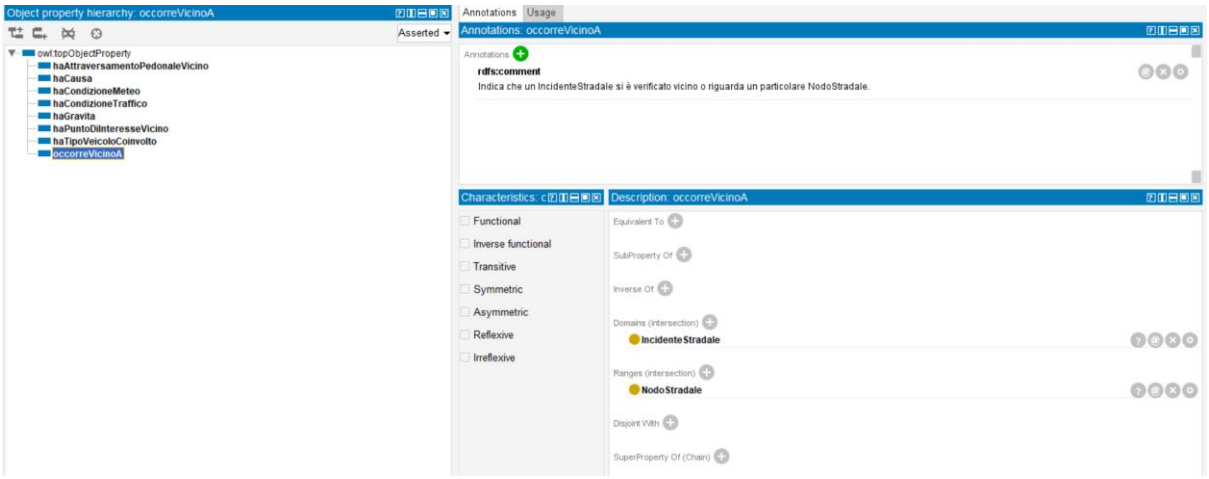
- **Classi Principali:**
  - **IncidenteStradale:** Il concetto centrale, che rappresenta un singolo evento di incidente.
  - **NodoStradale:** Rappresenta un punto specifico nella rete stradale, come un incrocio o un tratto di strada, dove possono verificarsi incidenti.
  - **PuntoDiInteresse:** Categorie di luoghi significativi nelle vicinanze di un nodo stradale (es. Scuola, Ospedale, Negozio).
  - **AttraversamentoPedonale:** Un tipo specifico di struttura pedonale.
  - **CausaIncidente, CondizioneMeteo, GravitaIncidente, TipoVeicoloCoinvolto:** Classi ausiliarie che categorizzano gli aspetti di un incidente



Classi dell'ontologia

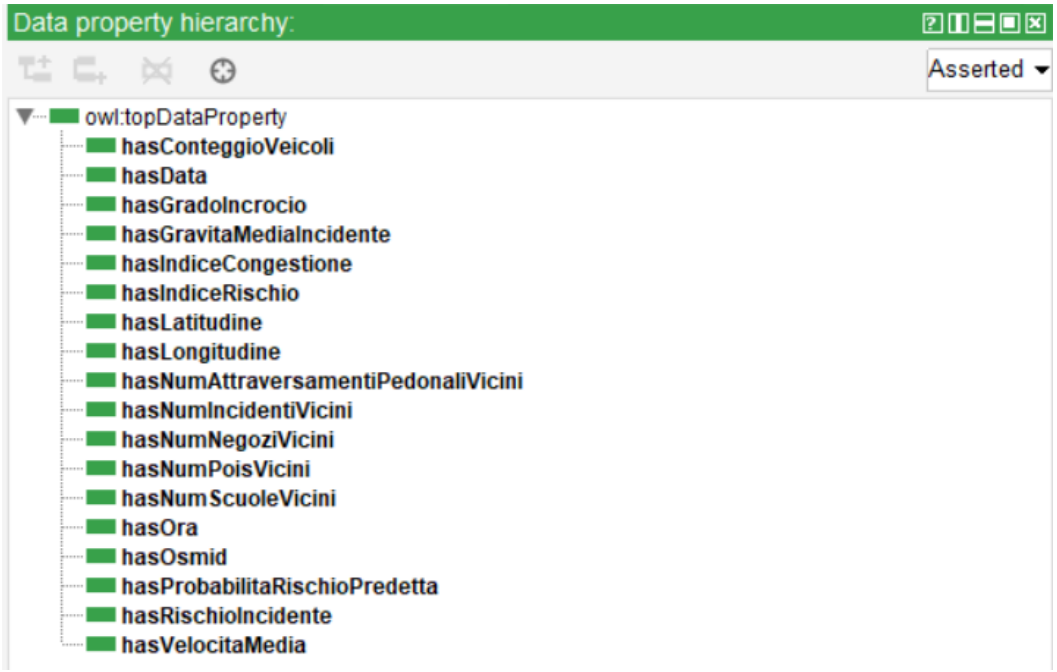
- **Relazioni** (Proprietà):
  - **Object Properties** (collegamenti tra Concetti):
    - **occorreVicinoA:** Collega un IncidenteStradale a un NodoStradale per indicare dove è avvenuto l'incidente.
    - **haCausa, haCondizioneMeteo, haGravita, haTipoVeicoloCoinvolto:** Collegano un IncidenteStradale alle rispettive categorie che lo descrivono.
    - **haPuntoDiInteresseVicino:** Collega un NodoStradale a un PuntoDiInteresse nelle sue vicinanze.
    - **haAttraversamentoPedonaleVicino:** Collega un NodoStradale a un AttraversamentoPedonale vicino.

Object Property

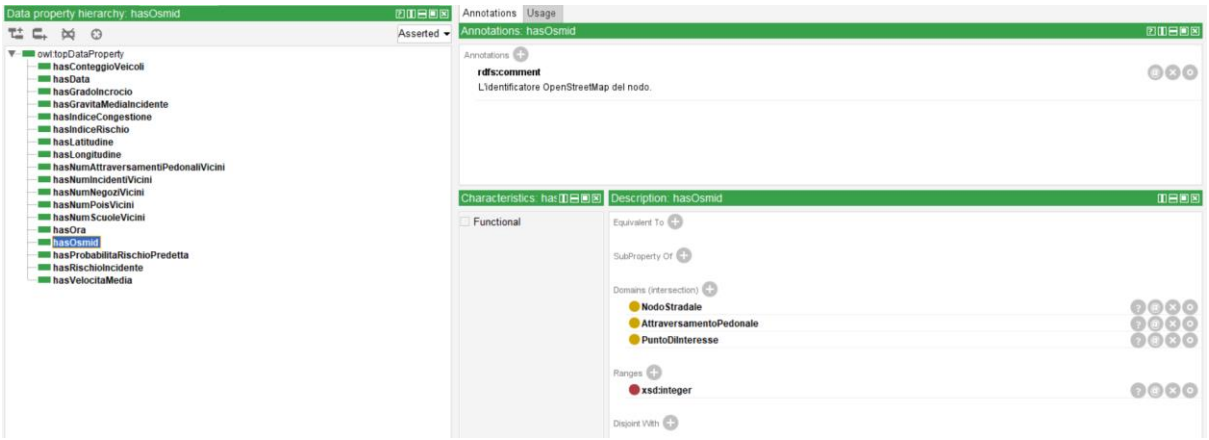


Struttura Object Property (es. occorreVicinoA)

- **Data Properties** (collegamenti a Valori):
  - **hasData, hasOra, hasLatitudine, hasLongitudine:** Dati temporali e spaziali per IncidenteStradale e NodoStradale (e PuntoDiInteresse).
  - **hasOsmid:** Un identificatore univoco (spesso da OpenStreetMap) per NodoStradale, AttraversamentoPedonale e PuntoDiInteresse.
  - **hasGradoIncrocio, hasNumAttraversamentiPedonaliVicini, hasIndiceRischio:** Metriche specifiche per descrivere un NodoStradale.

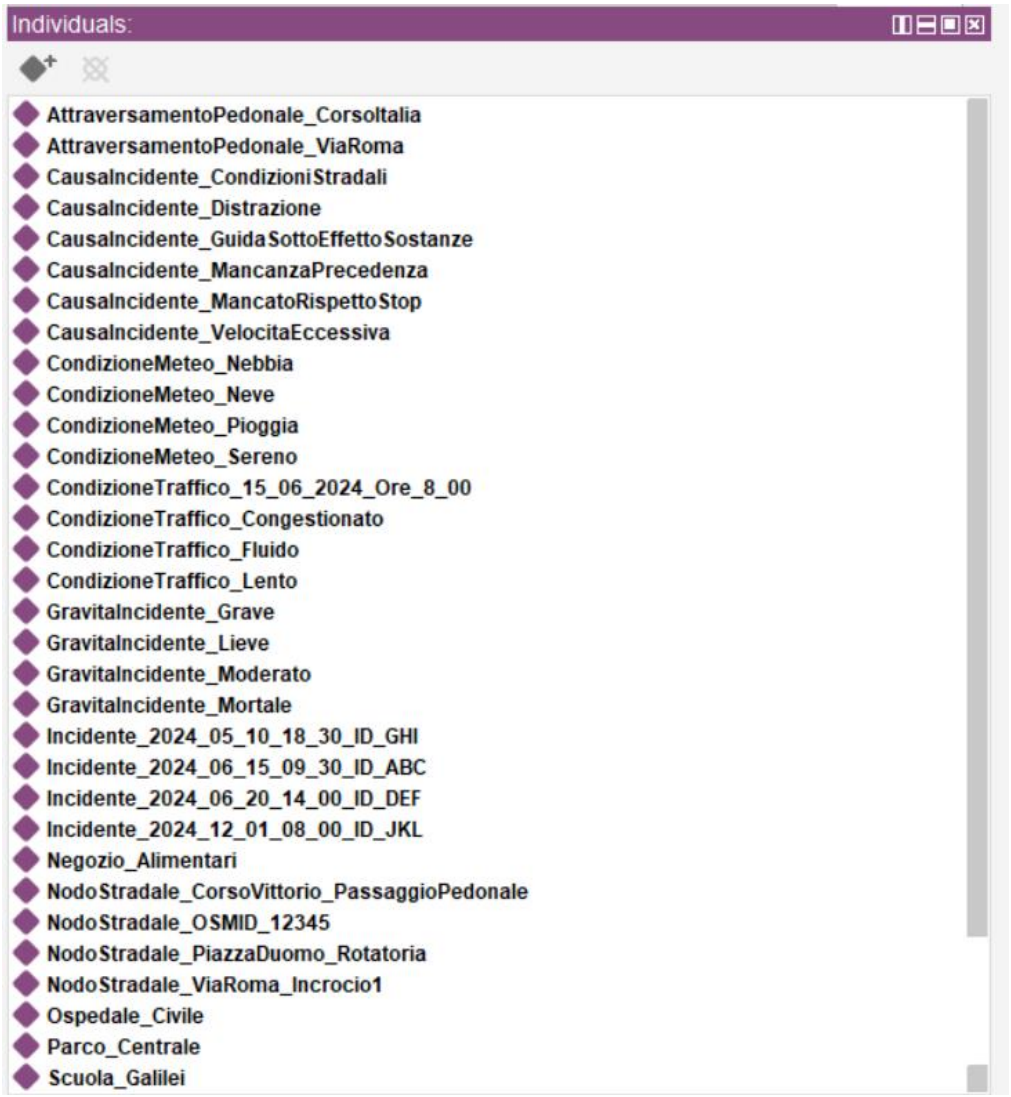


Data Property



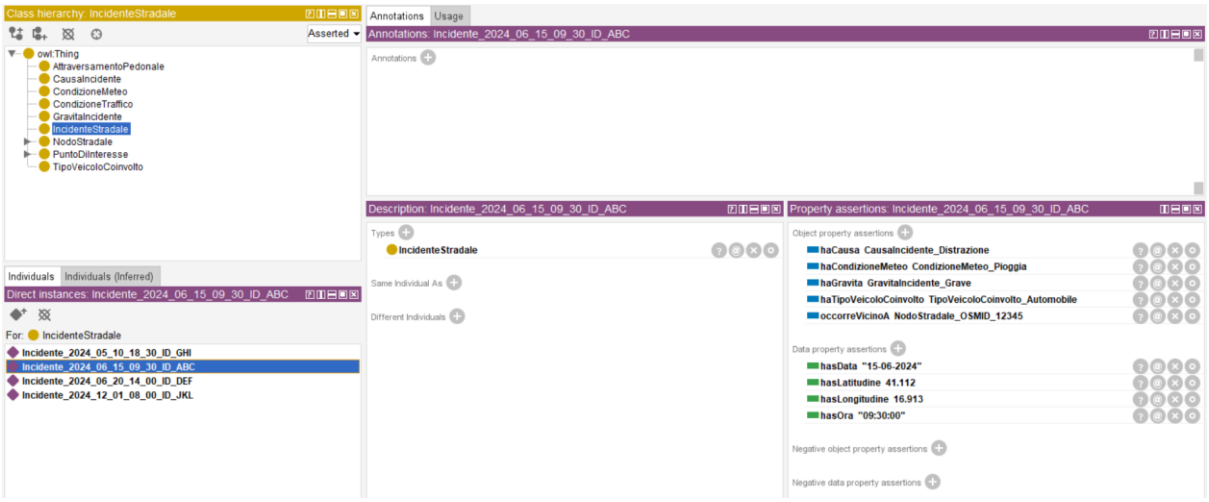
Struttura Data Property (es. hasOsmid)

- **Individui** (Dati Reali): Gli individui rappresentano le istanze concrete delle classi e sono la base su cui vengono applicate le proprietà e si costruisce la conoscenza effettiva. Nella nostra ontologia, gli individui sono i dati effettivi degli incidenti, dei nodi stradali, e dei punti di interesse.

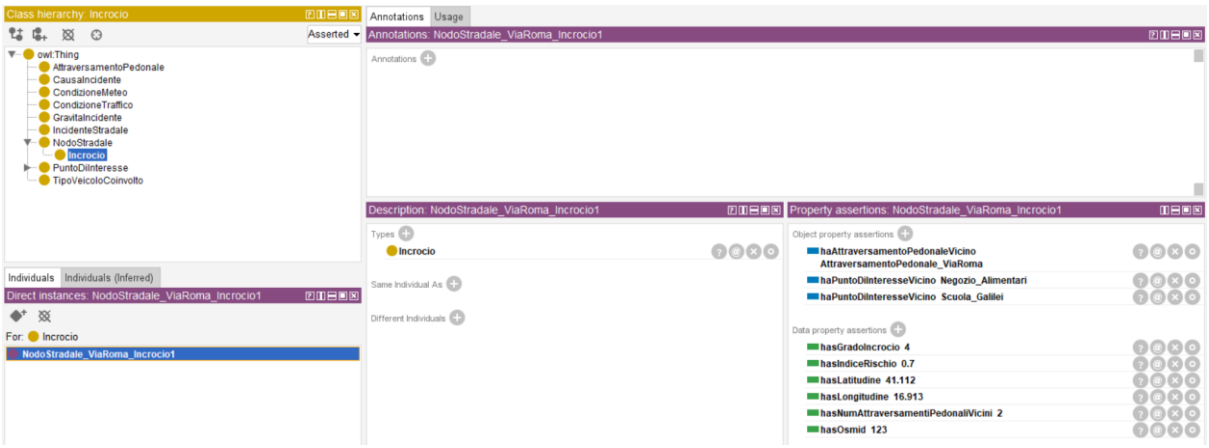


Es. Individui

- Alcuni esempi concreti:
  - **Incidente_2024_06_15_09_30_ID_ABC:** Un'istanza di IncidenteStradale, con dettagli specifici come la data ("15-06-2024”), l'ora ("09:30:00"), la causa (CausaIncidente_Distrazione), la gravità (GravitaIncidente_Grave), ecc.
  - **NodoStradale_ViaRoma_Incrocio1:** Un'istanza di Incrocio (sottoclasse di NodoStradale), con le sue coordinate geografiche (hasLatitudine, hasLongitudine), un ID OpenStreetMap (hasOsmid), e collegamenti a punti di interesse vicini (es. Scuola_Galilei).
  - **CausaIncidente_Distrazione:** Un'istanza della classe CausaIncidente, rappresentando una specifica ragione per un incidente.
  - **Scuola_Galilei:** Un'istanza di Scuola (sottoclasse di PuntoDiInteresse), con i suoi attributi come latitudine e longitudine.



Esempio di individuo con proprietà annesse



Esempio di individuo con proprietà annesse (1)

N.B: Visto l'elevato numero di record nel dataset (perlopiù talvolta generati a runtime in modo fittizio) non era possibile creare un individuo per ogni singola riga del dataset. Sono stati dunque creati alcuni individui rappresentativi per dimostrare l'ontologia e per testare alcune definizioni. Se si volesse popolarla completamente, si potrebbe usare uno script di programmazione (es. Python con librerie come *rdflib*) per mappare i dataset CSV/Excel all'ontologia. Per la fase di costruzione e dimostrazione, abbiamo ritenuto sufficiente qualche individuo manuale.

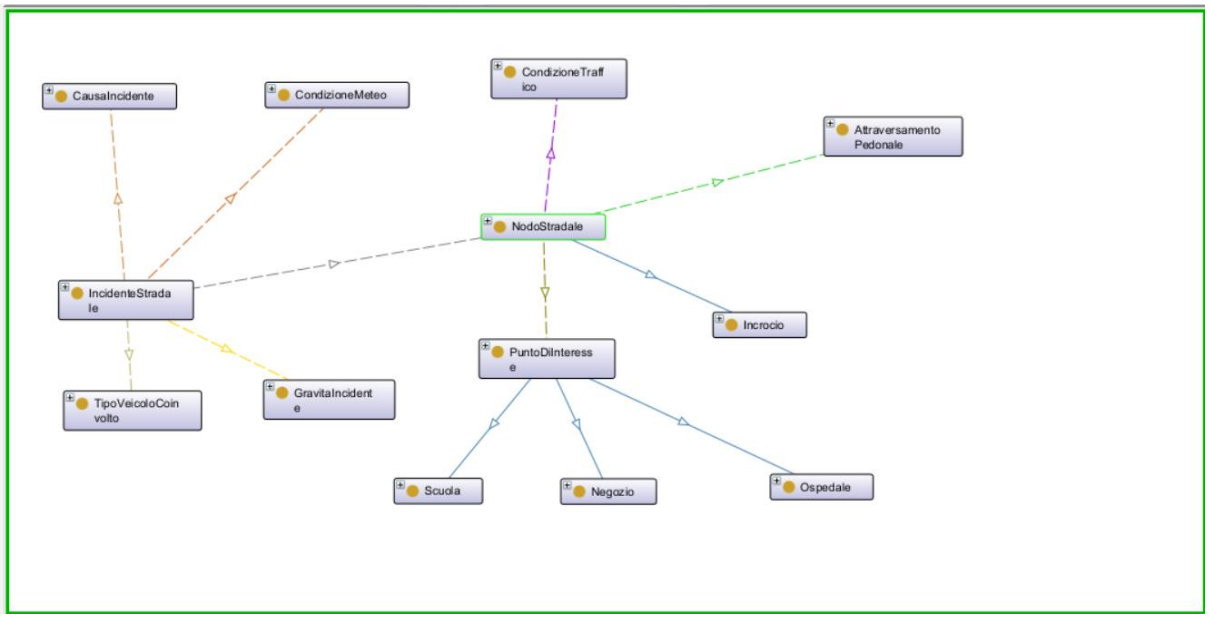


Immagine ontologia ad alto livello mediante OntoGraf

5.3 Esempi query

Nel contesto delle ontologie basate su OWL e RDF, le query sono il meccanismo principale per recuperare informazioni specifiche o per scoprire relazioni e pattern all'interno dei dati. Il linguaggio standard per interrogare i dati RDF e le ontologie OWL è **SPARQL** (*SPARQL Protocol and RDF Query Language*). Le **DL Query** (*Description Logic Queries*), invece, sono un modo potente per interrogare la struttura logica dell'ontologia piuttosto che gli individui specifici come fa principalmente SPARQL.

SPARQL query:

```
PREFIX pianificazione_urbana: <http://www.semanticweb.org/gaetano/ontologies/2025/6/pianificazioneUrbana#>

SELECT ?incidente ?causa
WHERE {
  ?incidente rdf:type pianificazione_urbana:IncidenteStradale .
  ?incidente pianificazione_urbana:haCausa ?causa .
}
```

incidente	causa
Incidente_2024_06_20_14_00_ID_DEF	CausaIncidente_VelocitaEccessiva
Incidente_2024_06_15_09_30_ID_ABC	CausaIncidente_Distrazione
Incidente_2024_05_10_18_30_ID_GHI	CausaIncidente_MancanzaPrecedenza
Incidente_2024_12_01_08_00_ID_JKL	CausaIncidente_CondizioniStradali

SPARQL query per trovare tutti gli incidenti e le loro cause

DL query:

Query (class expression)

IncidenteStradale

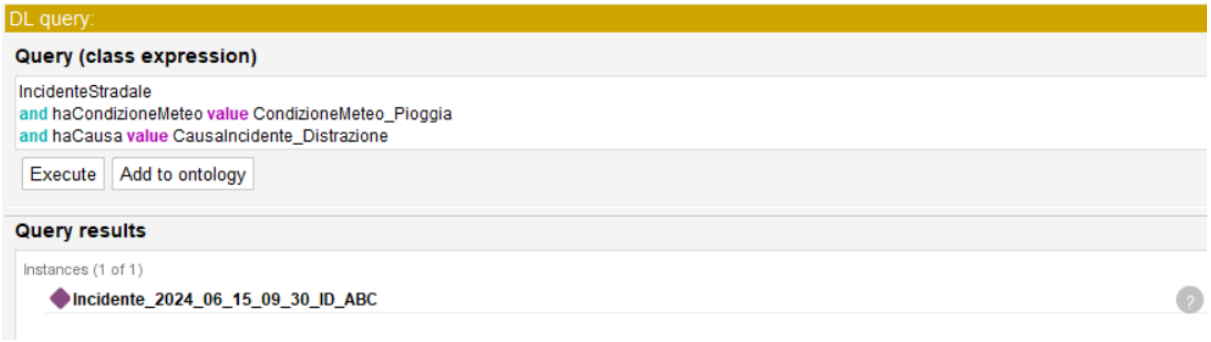
Execute | Add to ontology

Query results

Instances (4 of 4)

- Incidente_2024_05_10_18_30_ID_GHI
- Incidente_2024_06_15_09_30_ID_ABC
- Incidente_2024_06_20_14_00_ID_DEF
- Incidente_2024_12_01_08_00_ID_JKL

DL query per trovare gli individui che appartengono alla classe IncidenteStradale



DL query per trovare gli incidenti causati da pioggia e distrazione del conducente

6 Knowledge Base

Una **Knowledge Base** (KB), o Base di Conoscenza, è un tipo speciale di database che non si limita a memorizzare dati grezzi, ma organizza e rappresenta la conoscenza in un formato che può essere compreso, elaborato e ragionato da sistemi informatici. L'obiettivo principale di una KB è permettere a un sistema di "capire" il significato dei dati e di fare inferenze logiche, non solo di recuperare informazioni.

6.1 Popolamento KB

Il popolamento della Knowledge Base (KB) nel nostro progetto adotta un approccio fittizio e dinamico, differenziandosi dalla tradizionale pratica di caricare un intero dataset preesistente e statico. Questa scelta è dettata principalmente da esigenze di dimostrazione, test e flessibilità, piuttosto che dalla necessità di replicare un sistema su larga scala con dati reali e complessi.

6.2 Caricamento/Interrogazione KB

Il codice incapsulato nella classe *OntologyManager*, offre un'interfaccia Python per caricare, esplorare ed interrogare un'ontologia OWL/RDF, in questo caso specifica per la "Pianificazione Urbana". Utilizza la libreria *owlready2*, che permette di trattare le ontologie come oggetti Python, semplificando notevolmente l'interazione con dati semantici.

Permette, dunque, di:

- Caricare l'ontologia (file .rdf/.owl) in memoria, rendendola un oggetto Python.
- Esplorare la struttura dell'ontologia, permettendo di visualizzare le classi, le proprietà (che collegano entità o entità a valori) e le loro relazioni.
- Interrogare l'ontologia usando query SPARQL predefinite.
- Presentare un'interfaccia interattiva a riga di comando per scegliere ed eseguire queste query, visualizzando i risultati.

```
 ------------------------------------------------
 |  Inizio del Progetto di Pianificazione Urbana  |
 ------------------------------------------------

Seleziona un'opzione:
1: Interagisci con l'Ontologia
2: Procedi con l'analisi e la selezione della città
3: Esci dal Programma

Inserisci il numero dell'operazione desiderata: 1

Caricamento dell'ontologia...
Ontologia 'ontology/Ontologia_PianificazioneUrbana.rdf' caricata con successo.

Seleziona un'operazione sull'Ontologia:
1: Visualizzazione Classi
2: Visualizzazione Object properties
3: Visualizzazione Data properties
4: Esegui query predefinita
5: Torna al menu principale

Inserisci il numero dell'operazione desiderata:
```

Menu per interazione KB

```
Inserisci il numero dell'operazione desiderata: 4

--- Esegui Query Predefinite ---
1: Trova tutti gli individui di IncidenteStradale.
2: Trova incidenti causati da 'Distrazione' e la loro latitudine.
3: Trova tutti gli incidenti avvenuti in data odierna (2025-06-26).
4: Trova i Nodi Stradali e la loro longitudine.
5: Conta il numero totale di Incidenti Stradali.
6: Trova Nodi Stradali con incidenti e punti di interesse vicini.
7: Incidenti avvenuti di sera (dopo le 18:00) con gravità lieve.
8: Incidenti avvenuti di mattina (prima delle 12:00) con condizione meteo 'Neve'.
0: Torna indietro

Inserisci il numero della query da eseguire (o 0 per tornare indietro):
```

Esempi di query per interrogare la KB

Implicazioni per la KB:

- **KB Esemplare:** La KB risultante è una Knowledge Base esemplare, ovvero una rappresentazione ridotta ma semanticamente fedele di come i dati reali potrebbero essere modellati e interrogati.
- **Non esaustiva:** Non è una rappresentazione esaustiva di tutti i possibili incidenti, nodi o punti di interesse di una città reale, ma piuttosto un "mini-mondo" funzionale per validare la logica ontologica.
- **Verifica della Modellazione:** Questo approccio è particolarmente utile per verificare la correttezza della modellazione dell'ontologia e la validità delle query SPARQL su un set di dati controllato.

In sintesi, il popolamento della nostra KB si basa (spesso) sulla creazione mirata di dati principalmente fittizi, selezionando solo le informazioni essenziali per dimostrare le funzionalità dell'ontologia, piuttosto che replicare un intero e complesso scenario del mondo reale.

7 Conclusioni

Questo progetto ha affrontato la complessa sfida della predizione del rischio stradale, un tema di grande rilevanza sociale e di sicurezza pubblica. Attraverso un approccio basato sul Machine Learning, siamo riusciti a sviluppare un modello capace di identificare con buona accuratezza i nodi stradali a rischio di incidente, seppur basata (almeno nel nostro caso) su dati generati casualmente, anche se nel modo più accurato possibile in base alla città considerata.

Inoltre, in quanto progettato unicamente a scopo di studio, al momento è possibile selezionare unicamente 3 città (Terlizzi, Molfetta, Bari) per verificare il funzionamento.

7.1 Difficoltà riscontrate

L'assenza di un dataset preesistente e strutturato per la specifica applicazione ci ha portato a dover implementare delle funzioni in Python al fine di generare i dati di input in modo fittizio (seppur nel modo più accurato possibile). Sebbene questa soluzione abbia permesso di eseguire con successo l'applicazione, ha introdotto varie difficoltà, tra cui:

- **Mancanza di Realismo e Varietà:** La generazione fittizia ha reso difficile replicare la complessità e la varietà del mondo reale, limitando le combinazioni di relazioni (es. tra incidenti, cause e condizioni meteo) e la distribuzione statistica dei valori. Ciò poteva influire sulla robustezza dei test e sulla rappresentatività dei risultati delle query.
- **Complessità nel Debugging:** In caso di risultati inattesi, era spesso difficile distinguere se il problema risiedesse nella logica di generazione dei dati fittizi, o in altre parti relative al processing, non avendo un "ground truth" (dati reali verificabili) con cui confrontarsi.

Inoltre, relativamente all'utilizzo di Protégé, abbiamo riscontrato alcune difficoltà, tra cui:

- **Errori di Parsing del File RDF:** Il problema più critico è stato la presenza di errori di sintassi nel file RDF dell'ontologia, come il doppio cancelletto (##) negli URI delle proprietà e altri errori di formattazione XML. Questi errori impedivano a Protégé di caricare correttamente l'ontologia, bloccando di fatto qualsiasi operazione successiva e rendendo i ragionamenti e le query inaffidabili.
- **Inconsistenza Logica dell'Ontologia:** Anche quando il file RDF veniva caricato, l'ontologia risultava spesso inconsistente. Questo era dovuto principalmente a:
    - **Domini delle Proprietà Definiti Erroneamente:** Un problema significativo è stato l'errata definizione del dominio delle proprietà hasLatitudine e hasLongitudine come IncidenteStradale. Ciò comportava che qualsiasi individuo con un valore per queste proprietà venisse inferito erroneamente come IncidenteStradale (es. Negozio_Alimentari o Scuola_Galilei erano classificati come incidenti).
    - **Mancanza o Errata Specificazione dei Datatype:** Per le proprietà con valori numerici, di data o ora (es. hasLatitudine, hasData, hasOra), l'assenza o la formattazione errata dell'attributo rdf:datatype nel file RDF impediva al ragionatore di interpretare correttamente i valori, contribuendo all'inconsistenza.

7.2 Sviluppi Futuri

Il potenziale di espansione e miglioramento di questo progetto è vasto. Tra i principali sviluppi futuri che potrebbero essere intrapresi, includiamo:

- **Integrazione di Dati Dinamici e Temporali:** Attualmente, il modello si basa principalmente su feature statiche o medie. L'integrazione di dati in tempo reale o quasi reale (es. dati GPS anonimi, flusso veicolare in tempo reale, condizioni meteo attuali) potrebbe permettere la previsione del rischio non solo per un nodo, ma per una specifica fascia oraria o condizione ambientale. Questo trasformerebbe il modello da predittivo a livello di nodo a predittivo a livello di contesto dinamico.
- **Feature Engineering Avanzato:** Esplorare ulteriori feature più complesse, come:
    - **Interazioni tra feature:** Creare nuove feature combinando quelle esistenti (es. rapporto tra volume di traffico e numero di corsie).
    - **Analisi dei pattern di manovra:** Se disponibili, dati sui tipi di manovre effettuate negli incroci.
    - **Vicinanza a servizi di emergenza:** La distanza dai pronto soccorso o vigili del fuoco.
- **Validazione su Nuovi Dati e Monitoraggio Continuo:**
    - Eseguire test approfonditi su set di dati provenienti da periodi futuri o aree geografiche diverse per confermare la generalizzabilità del modello.
    - Implementare un sistema di monitoraggio continuo per tracciare le prestazioni del modello nel tempo e identificare eventuali "drift" dei dati (ovvero se la distribuzione dei dati di input che il modello riceve inizia a cambiare rispetto alla distribuzione dei dati su cui il modello è stato addestrato) o dei pattern di rischio, permettendo un riaddestramento tempestivo.
- **Applicazioni Pratiche e Interfaccia Utente:** Sviluppare un'interfaccia utente (web app o dashboard) che permetta alle autorità competenti di visualizzare i nodi a rischio su una mappa, applicare filtri e ottenere report, facilitando l'adozione e l'utilizzo operativo del modello.
- **Impostare dinamicamente la soglia di priorità:** Attualmente, se lo score di rischio di un nodo stradale (compreso tra 0 e 1) è >= 0.8, automaticamente verrà associata la priorità alta. Tuttavia, si potrebbe limare questo dettaglio, magari considerando anche la popolazione (se una città è piccola, allora ci saranno meno incidenti in media, rispetto a una grande città).

Questi sviluppi permetterebbero di trasformare il prototipo attuale in una soluzione ancora più dinamica, precisa e utile per la gestione e la sicurezza della rete stradale.