Relazione Tecnica per il Concorso "Sii Saggio, Guida Sicuro" – XI Edizione 2024-2025

Introduzione

Il progetto presentato si propone di affrontare il tema della sicurezza stradale attraverso un approccio innovativo che combina **analisi predittiva** e **collaborazione con le istituzioni locali**. Partendo dai dati grezzi forniti dal Comando di Polizia municipale di Cava de' Tirreni, abbiamo sviluppato un sistema basato sull'intelligenza artificiale in grado di identificare le aree a maggior rischio di incidenti e suggerire interventi mirati. Questo lavoro si allinea perfettamente con gli obiettivi del bando, poiché non solo analizza il problema, ma offre anche **uno** strumento concreto per la sua mitigazione.

Digitalizzazione e Preparazione dei Dati

Il punto di partenza del progetto è stato il **file Excel** contenente i dati degli incidenti stradali registrati nel comune di Cava de' Tirreni. Questi dati, originariamente raccolti in forma cartacea, sono stati digitalizzati mantenendo la struttura originale dei moduli compilati dalla polizia municipale.

Struttura del Dataset

Il dataset include informazioni dettagliate su:

- **Localizzazione**: Coordinate geografiche (latitudine e longitudine) per ogni incidente.
- **Temporali**: Data e ora dell'incidente, con possibilità di estrarre ulteriori dettagli come il giorno della settimana.
- **Contestuali**: Tipo di strada, condizioni meteorologiche, illuminazione, numero di veicoli coinvolti e natura dell'incidente.

Processo di Pulizia e Preparazione

Sebbene il file Excel originale non sia stato modificato direttamente, il codice Python sviluppato effettua una **pulizia dinamica** dei dati al momento della loro elaborazione. Questo approccio garantisce flessibilità e riproducibilità, mantenendo intatto il dataset originale. Le operazioni principali includono:

1. **Rimozione di dati incompleti**: Le righe prive di coordinate geografiche (essenziali per l'analisi) vengono escluse automaticamente durante l'esecuzione del codice.
2. **Conversione dei tipi di dati**: Le colonne numeriche (es. "N° veicoli coinvolti") vengono convertite in valori float, mentre i campi testuali (es. "Condizioni meteo") vengono uniformati sostituendo i valori mancanti con "Sconosciuto".
3. ***Feature engineering***: Vengono generate nuove variabili, come il giorno della settimana e un flag per identificare i weekend, per arricchire l'analisi.

Questa fase è cruciale perché trasforma i dati grezzi in un formato adatto all'addestramento del modello, senza alterare il file originale.

Il Modello Predittivo: Architettura e Funzionamento

Il cuore del progetto è una *rete neurale* sviluppata con *TensorFlow/Keras*, progettata per prevedere le coordinate geografiche degli incidenti sulla base di variabili descrittive. La scelta di una rete neurale è particolarmente adatta a questo tipo di problema, poiché permette di catturare relazioni complesse e non lineari tra i diversi fattori che contribuiscono al verificarsi di incidenti, relazioni che spesso sfuggono ai modelli statistici tradizionali.

Selezione delle Feature

Le feature utilizzate includono:

- **Variabili temporali**: Giorno della settimana, ora del giorno, weekend. Queste variabili permettono di cogliere le fluttuazioni nella frequenza degli incidenti legate ai ritmi della vita quotidiana.
- **Variabili contestuali**: Tipo di strada, condizioni meteo, illuminazione, pavimentazione. Questi fattori ambientali e infrastrutturali hanno un impatto diretto sulla sicurezza stradale, influenzando sia la visibilità che l'aderenza del veicolo alla strada.
- **Variabili incidente**: Numero di veicoli coinvolti, natura dell'incidente.

Pre Elaborazione dei Dati

Prima di essere forniti alla rete neurale, i dati subiscono una **standardizzazione** (con `StandardScaler` di scikit-learn) per garantire che tutte le feature contribuiscano equamente al modello. Questo passaggio è essenziale per evitare che variabili con scale diverse (es. "ora del giorno" vs. "numero di veicoli") dominino il processo di apprendimento..

Architettura della Rete Neurale

La rete è composta da:

1. **Livello di input**: Accetta le 10 feature selezionate.
2. **Livelli nascosti**:
    a. Un primo strato con 128 neuroni e attivazione ReLU, seguito da un *dropout* del 20% per ridurre *l'overfitting*.
    b. Un secondo strato con 64 neuroni e ReLU.
    c. Un terzo strato con 32 neuroni e ReLU.
3. **Livello di output**: 2 neuroni (per latitudine e longitudine) senza funzione di attivazione.

La rete è ottimizzata con l'algoritmo **Adam** e la **loss** è calcolata come errore quadratico medio (***MSE***), una scelta standard per problemi di regressione.

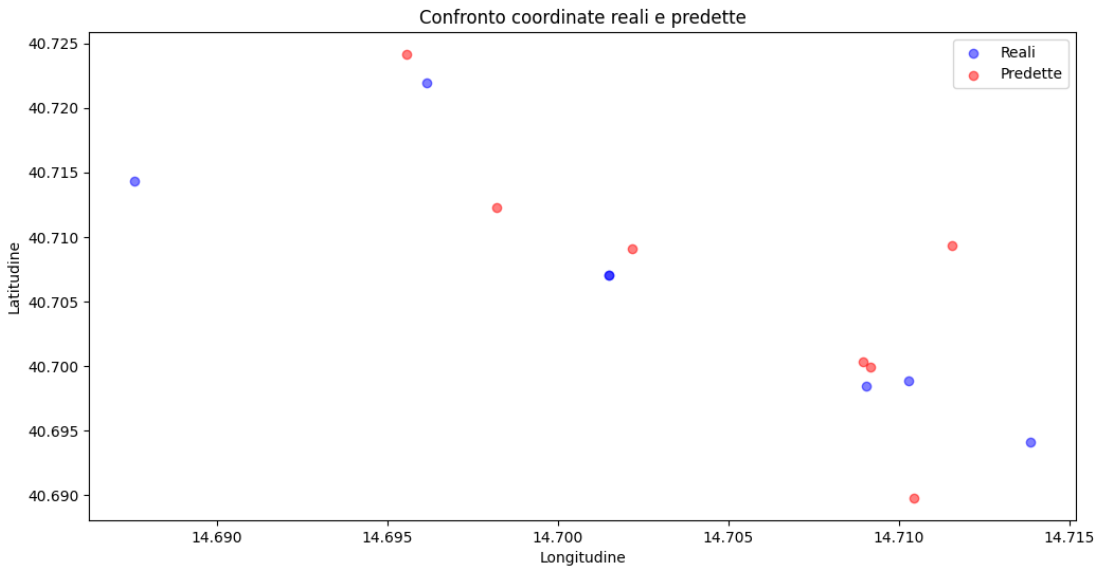
Addestramento e Validazione

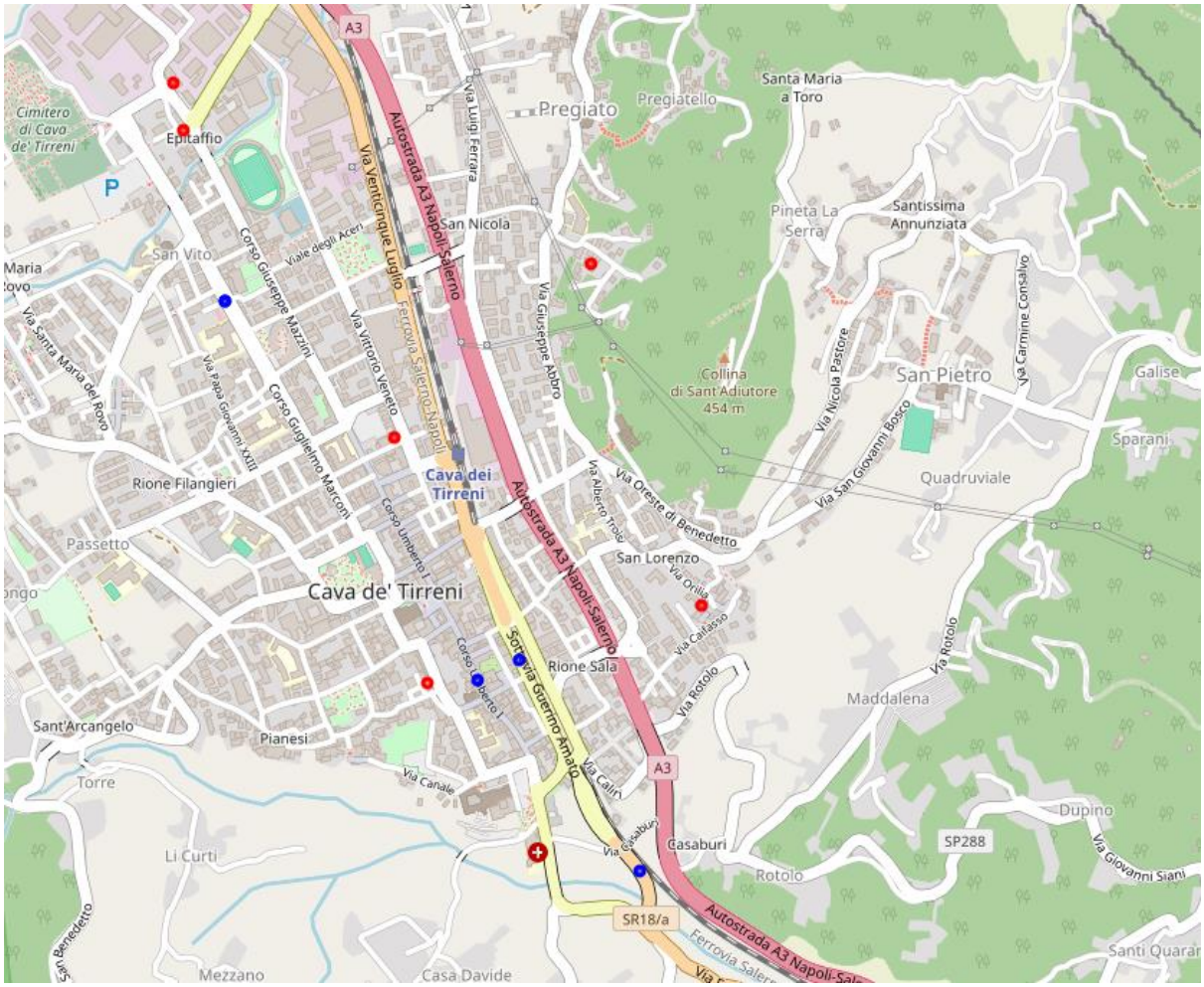
Il modello è stato addestrato su un subset dei dati (80%) e validato sul restante 20%. Durante l'addestramento, sono state monitorate sia la loss di training che quella di validazione per evitare overfitting. Il valore di MSE ottenuto quantifica l'errore medio nelle predizioni delle coordinate, fornendo una misura oggettiva dell'accuratezza del modello. Un MSE basso indica che le predizioni sono vicine alle coordinate reali degli incidenti. Il valore di MSE ottenuto è: *0.00016655225085544927*

Risultati e Visualizzazione

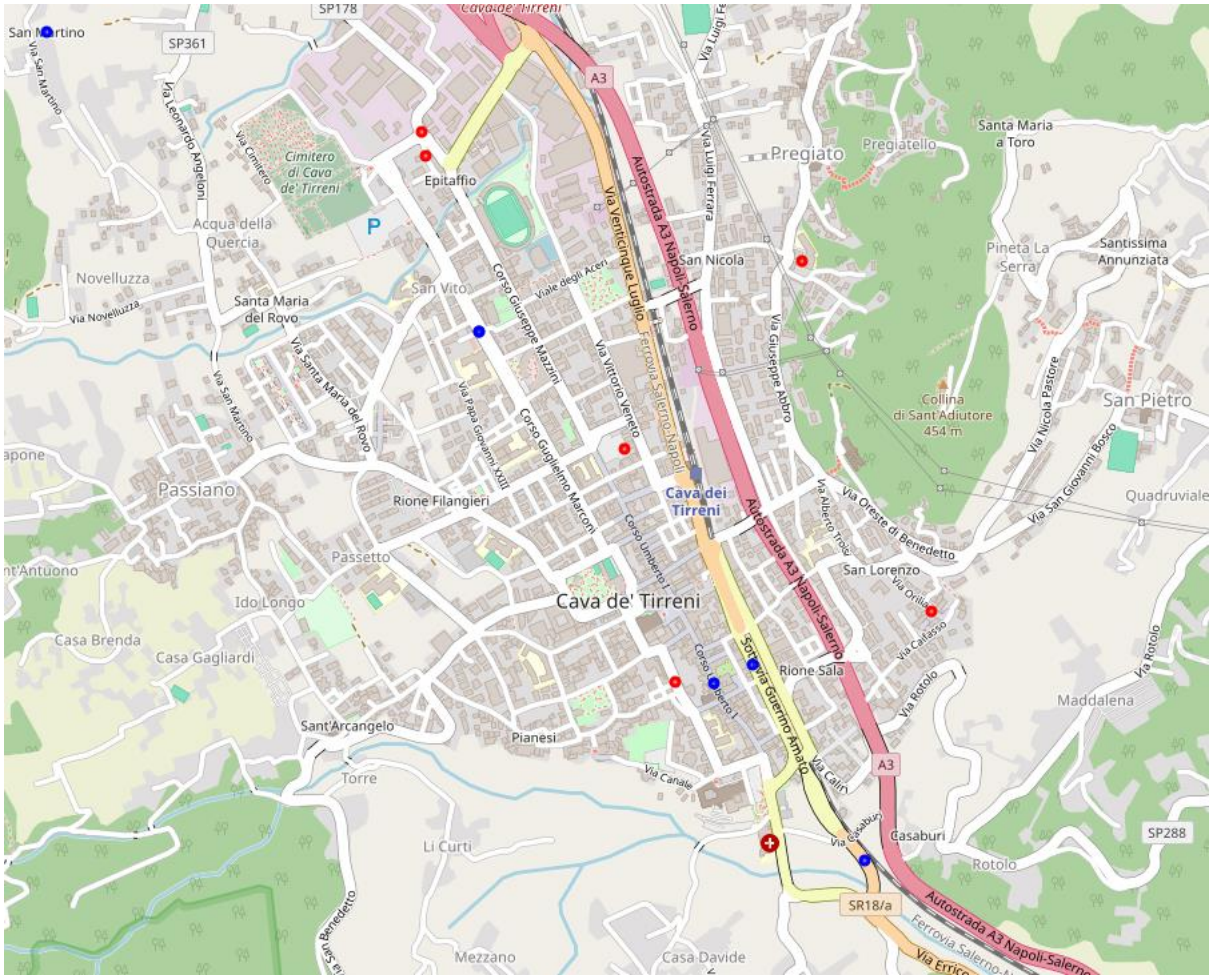
Le predizioni del modello sono state confrontate con i dati reali attraverso due modalità complementari:

1. **Grafico scatter**: I punti blu rappresentano gli incidenti reali, mentre i punti rossi mostrano le predizioni del modello.
2. **Mappa interattiva con *Folium***: Offre una rappresentazione geospaziale immediatamente interpretabile, dove gli incidenti reali e predetti sono visualizzati come marcatori su una mappa reale di Cava de' Tirreni. Questa visualizzazione facilita l'identificazione delle zone a rischio e permette anche di valutare l'accuratezza spaziale del modello nel contesto reale del territorio comunale. Può verificarsi che le coordinate predette non corrispondano ad una strada del territorio. A tale scopo è stato integrato un ***OSRM(open source routing machine)***, ossia un motore open-source per il calcolo di percorsi ottimizzati su reti stradali. Utilizza algoritmi di routing come *Dijkstra* e *Contraction Hierarchies* per determinare il percorso più breve o veloce tra punti geografici. Offre anche il servizio *nearest* per effettuare il match di coordinate imprecise sulla strada più vicina. L'API utilizzata è la seguente: *link*.





(mappa senza OSRM)



(mappa con OSRM)

Future Implementazioni

1. **Salvataggio e utilizzo del modello predittivo**: Attualmente il progetto include una rete neurale addestrata su dati storici. Il prossimo passo sarà salvare il modello (es. in formato .h5 con TensorFlow/Keras) per poter effettuare previsioni in tempo reale.
2.

Ringraziamenti e collaborazioni istituzionali

La realizzazione del presente progetto è stata possibile grazie al prezioso contributo delle seguenti istituzioni e professionisti:

- **Alla professoressa di italiano e storia Antonia Silvestri dell'IIS Della Corte-VanVitelli** per il supporto organizzativo e per averci accompagnato durante gli incontri presso il comando della polizia municipale.
- **All'Assessore all'istruzione Lorena Iuliano del Comune di Cava de' Tirreni**, per aver facilitato il dialogo con le istituzioni e messo a disposizione i contatti istituzionali;
- **Al Maggiore Michele Lamberti del Comando della Polizia municipale**, per l'accesso ai dati e la consulenza tecnica sugli aspetti operativi;
- **All'ingegnere Claudio Troisi e all'Architetto Lorena Pisapia redattori del piano del traffico e del piano della sicurezza stradale urbana**, per la guida specialistica nell'interpretazione dei documenti dei sinistri stradali e l'analisi del contesto urbano.

Un particolare riconoscimento va al Comando di Polizia municipale per la trasparenza nella condivisione dei dati storici, dimostrando come la collaborazione tra scuole e istituzioni possa generare strumenti concreti per la sicurezza cittadina.

Conclusioni

Questo progetto dimostra concretamente come l'intelligenza artificiale possa essere utilizzata per affrontare problemi reali di sicurezza stradale. La collaborazione con la polizia municipale e con il Comune ha garantito la qualità dei dati.

Progetto dell'IIS Della Corte-VanVitelli Cava Dei Tirreni(SA)

Team di Sviluppo:

Docente di riferimento: Amendola Maria Emilia

Sviluppo rete neurale:

- Ostoni Christian: rete neurale ed OSRM
- Luciano Vincenzo: integrazione mappa con folium

Digitalizzazione dei dati:

- Manzo Pasquale
- Ferrigno Federico
- Tortora Cristiano
- Luciano Vincenzo

Glossario

Feature Engineering
Processo di creazione o trasformazione di variabili (feature) per migliorare la capacità predittiva di un modello.

StandardScaler
Tecnica di standardizzazione dei dati che porta ogni variabile ad avere media 0 utile per l'addestramento di modelli di machine learning.

Rete Neurale
Modello matematico ispirato al funzionamento del cervello umano, composto da nodi (neuroni artificiali) collegati tra loro.

TensorFlow/Keras Librerie
open-source di machine learning in Python, utilizzate per costruire e addestrare reti neurali.

Dropout
Tecnica di regolarizzazione che consiste nel "spegnere" casualmente alcuni neuroni durante l'addestramento della rete neurale per ridurre il rischio di overfitting.

Overfitting
Fenomeno in cui un modello impara troppo bene i dati di addestramento, perdendo la capacità di generalizzare su dati nuovi.

MSE (Mean Squared Error)
Indicatore che misura la media degli errori al quadrato tra valori predetti e valori reali. Più è basso, migliore è la qualità delle predizioni.

Folium
Libreria Python che permette di creare mappe interattive, utili per visualizzare dati geospaziali come la posizione degli incidenti.

OSRM (Open Source Routing Machine)
Motore open-source per il calcolo di percorsi ottimizzati su reti stradali. Viene usato anche per correggere coordinate geografiche imprecise trovando la strada più vicina.

Dijkstra
Algoritmo che trova il percorso più breve tra due punti su un grafo (rete di strade, ad esempio).

Contraction Hierarchies
Tecnica di ottimizzazione che accelera il calcolo dei percorsi più brevi all'interno di reti molto grandi.

File allegati

- **Programma in python:**
  - *CoorMatch.py*: Contiene la classe per effetturare la richiesta API al servizio di OSRM per ottenere il match delle coordinate(*servizio nearest di OSRM)* su una strada concreta
  - *GetFilePath.py*: Contiene la classe per creare la GUI che consente di selezionare il file excel da analizzare
  - *main_with_maps.py*: E' il file principale che contiene la rete neurale e richiama le funzioni delle classi sopracitate.
- *previsore.exe*: File eseguibile della rete neurale. Genera nella cartella in cui è eseguito due pagine html:
  - *mappa_incidenti_with_match.html*: Contiene la mappa generata tramite *folium*. Le coordinate predette che vengono mostrate sono aggiustate tramite il servizio di OSRM.
  - *mappa_incidenti_without_match.html*: Contiene la mappa generata tramite *folium*. Le coordinate predette sono quelle "*orginali"* senza correzioni effettuate dall'OSRM.

- *dati_incidenti.xlsx*: E’ il file excel che contiene i dati digitalizzati dei sinistri stradali. Quando viene eseguito l’eseguibile è necessario selezionare questo file.
- *video_esplicativo.mkv*: E’ un video esplicativo che mostra come mandare in esecuzione l’eseguibile