

UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione



TESI DI LAUREA

**Analisi di dati su incidenti stradali: processi di elaborazione,
visualizzazione e previsione tramite Oracle APEX e Python**

**Analysis of road accident data: processing, visualization, and
forecasting using Oracle APEX and Python**

Relatore

Prof. Domenico Ursino

Candidata

Alessia Capancioni

Correlatore

Michele Marchetti

What I cannot create, I do not understand.

Richard Feynman

Sommario

La tesi affronta il tema dell'analisi e della visualizzazione dei dati sugli incidenti stradali in ambito urbano. A partire da un dataset relativo agli incidenti avvenuti nella città di Chicago, è stato inizialmente condotto un lavoro di valutazione della qualità dei dati, di modellazione relazionale in Oracle Database e di definizione delle viste dedicate alla pulizia, alla formattazione e all'aggregazione delle informazioni. Su queste basi, è stato sviluppato un applicativo web in Oracle APEX, organizzato in pagine con report e form, per la parte gestionale, e in pagine con dashboard tematiche filtrabili, per la visualizzazione facilitata e interattiva delle informazioni. In una fase successiva, i dati temporali degli incidenti sono stati impiegati per costruire un modello di analisi predittiva in un ambiente Python locale. Dopo un'introduzione teorica sui principali concetti di machine learning e di regressione su serie storiche, è stato implementato e valutato un modello di Random Forest Regressor, della libreria scikit-learn, per prevedere il conteggio settimanale degli incidenti. Nonostante i limiti dovuti ai vincoli dell'ambiente cloud e alle variabili disponibili, il lavoro mostra come sia possibile trasformare un dataset grezzo in un sistema integrato di gestione e visualizzazione dei dati, arricchito con analisi descrittive e predittive.

Keyword: Road Accident Data, Oracle Database, Oracle APEX, Business Intelligence, Data Visualization, Dashboard, Time Series, Machine Learning, Forecasting, Random Forest.

Introduzione	1
1 Introduzione ad Oracle APEX	3
1.1 Introduzione e contesto	3
1.1.1 Che cos'è Oracle APEX	4
1.2 Stack tecnologico	5
1.2.1 Oracle Database	5
1.2.2 ORDS	6
1.2.3 Architettura a tre livelli	7
1.2.4 Opzioni di installazione	7
1.3 OCI e Autonomous Database	8
1.3.1 Oracle Cloud Infrastructure	8
1.3.2 Autonomous Database	10
1.3.3 APEX con Autonomous Database	10
1.4 Ambienti e strumenti di Oracle APEX	11
1.4.1 Workspace	12
1.4.2 SQL Workshop	12
1.4.3 App Builder	15
1.5 Collegamento al caso di studio	17
1.5.1 Gestionale e reporting	17
2 Descrizione dei dati di partenza e attività di pre-processing	19
2.1 Dataset	19
2.1.1 Scelta e struttura del dataset	19
2.2 Metodi di elaborazione dei dati	20
2.2.1 Estrazione e caricamento	21
2.2.2 Normalizzazione e tabelle di lookup	23
2.2.3 Pulizia e vista di raggruppamento	27
2.3 Applicativo APEX	29
2.3.1 Report interattivi delle tabelle di lookup	30
2.3.2 Griglia interattiva degli incidenti stradali	31
2.3.3 Home Page e menù di navigazione	32

3 Progettazione e implementazione delle dashboard di base	34
3.1 Obiettivi di analisi e criteri di design	34
3.1.1 Data analytics e DDDM	34
3.1.2 Domande di analisi	35
3.1.3 Linee guida per la visualizzazione dei dati	35
3.2 Componenti comuni	37
3.2.1 Sorgente dei dati e mappatura delle colonne	37
3.2.2 Pattern di pagina	38
3.3 Dashboard tematiche interattive	39
3.3.1 Dashboard 1: Analisi temporale	39
3.3.2 Dashboard 2: Condizioni ambientali e stradali	43
3.3.3 Dashboard 3: Cause e dinamiche	46
3.3.4 Dashboard 4: Analisi geografica	48
3.3.5 Dashboard 5: Esiti e severità	49
4 Progettazione e implementazione dei grafici avanzati	52
4.1 Introduzione al machine learning	52
4.1.1 Metodi di machine learning	53
4.1.2 Supervised Learning	54
4.2 Serie storiche	55
4.2.1 Componenti principali	55
4.2.2 Analisi descrittiva e analisi predittiva	56
4.3 Addestramento e valutazione del modello	56
4.3.1 Suddivisione temporale dei dati	57
4.3.2 Metriche di errore	57
4.4 Realizzazione dei grafici avanzati con Python	57
4.4.1 Ambiente di analisi e strumenti	58
4.4.2 Preparazione dei dati e costruzione della serie storica settimanale	59
4.4.3 Problema supervisionato e modello di regressione	61
4.4.4 Valutazione del modello e generazione dei grafici	65
5 Discussione in merito al lavoro svolto	72
5.1 Valutazione complessiva, punti di forza e criticità	72
6 Conclusioni	75
Bibliografia	76
Sitografia	80

Elenco delle figure

1.1	Caratteristiche di Oracle APEX	5
1.2	Elenco delle funzionalità di Database Actions offerte da ORDS	6
1.3	Architettura a tre livelli di Oracle APEX	7
1.4	Opzioni di installazione dello stack tecnologico	8
1.5	Infrastruttura completa del cloud di Oracle	9
1.6	Servizi dell'Oracle Cloud Free Tier e APEX su Autonomous Database	10
1.7	Pagina dell'applicativo contenente una griglia interattiva con tutti i record degli incidenti stradali	11
1.8	Home page dell'area di lavoro APEX	12
1.9	Pagina di SQL Workshop con i suoi strumenti	13
1.10	Visualizzazione della tabella A01_TRAFFIC_ACCIDENTS nel Browser Oggetti	14
1.11	Scrittura ed esecuzione della query nell'editor di Comandi SQL per recuperare i dati relativi alle condizioni meteo degli incidenti stradali	14
1.12	Pagina dell'App Builder con tutti i suoi strumenti	16
1.13	Modale di selezione della tipologia di pagina da creare	16
1.14	Page Designer che mostra la struttura e l'editor di modifica della pagina numero 41 dell'applicazione	17
1.15	Griglia interattiva con form per l'inserimento e la modifica dei record della vista sugli incidenti stradali	18
1.16	Esempio di dashboard tematica con KPI e grafici interattivi	18
2.1	Struttura e valutazione del dataset preso dal sito Kaggle	20
2.2	Processo di ETL	21
2.3	Processo di ELT	21
2.4	Modifica manuale del parsing automatico delle colonne del dataset	22
2.5	Codice DDL di creazione della tabella A01_TRAFFIC_ACCIDENTS (prima parte)	22
2.6	Codice DDL di creazione della tabella A01_TRAFFIC_ACCIDENTS (seconda parte)	23
2.7	Schema esemplificativo di relazione matematica	23
2.8	Relazione nel modello relazionale, con attributi (colonne) e record (righe) . .	24
2.9	Esempio di normalizzazione in 1NF, con tabella dominio e tabella di collegamento N:N	25
2.10	Esempio di creazione automatica di una tabella di lookup	26
2.11	Contenuto della tabella principale dopo la creazione delle tabelle di lookup .	27
2.12	Esempio di raggruppamento dei valori per l'attributo first_crash_type .	28

2.13 Creazione del bucket per l'attributo <code>lane_cnt</code> e gestione dei valori non validi	28
2.14 Operazioni di TRIM e conversione in stringhe dei valori degli attributi temporali per il loro ordinamento e utilizzo nelle etichette dei grafici	29
2.15 Elenco di tutte le operazioni di left join effettuate tra la tabella principale e le tabelle di lookup durante la creazione della vista	29
2.16 a) Riquadro di sinistra del Page Designer con la struttura del report. b) Visualizzazione del report all'interno dell'applicativo. c) Form di inserimento di un nuovo record	30
2.17 Griglia interattiva della vista, con l'elenco di tutte le sue azioni	31
2.18 Filtri, aggregazioni e conteggi nella griglia interattiva	31
2.19 Home Page dell'applicativo con menù di navigazione	32
 3.1 Schema di tutte le tipologie di grafici suddivisi per macrocategorie	37
3.2 Sezioni del Page Designer con la query sorgente e il mapping delle colonne per il grafico <i>Crashes by Month and Year</i> della Dashboard 1	38
3.3 Pagina dell'applicativo APEX contenente la Dashboard 1 sull'analisi temporale	40
3.4 Query sorgente per le card del report classico dei KPI della Dashboard 1	41
3.5 Risultato di una singola query per un KPI	41
3.6 Apertura della pagina modale con la griglia degli incidenti pre-filtrata	41
3.7 Query sorgente del grafico <i>Cumulative Crashes by Year</i>	42
3.8 Query sorgente del grafico <i>Year Over Year Crash Count Change</i>	43
3.9 Pagina dell'applicativo APEX contenente la Dashboard 2 sulle condizioni ambientali e stradali	44
3.10 Query sorgente del grafico <i>Crash Severity by Critical Weather Conditions</i>	45
3.11 Query sorgente del grafico <i>Crashes by Lane Count and Top Road Defects</i>	45
3.12 Pagina dell'applicativo APEX contenente la Dashboard 3 sulle cause e le dinamiche	46
3.13 Visualizzazione dinamica dei dati nel grafico <i>Crash Type</i> tramite legenda interattiva	47
3.14 Pagina dell'applicativo APEX contenente la Dashboard 4 sull'analisi geografica	48
3.15 a) Struttura gerarchica dei componenti della <i>Crashes Map</i> , visualizzata nel riquadro sinistro del Page Designer). b) Mapping delle coordinate per la rappresentazione dei singoli incidenti nella mappa. c) Query sorgente per i tre layer	49
3.16 Pagina dell'applicativo APEX contenente la Dashboard 5 su esiti e severità	50
3.17 a) Struttura gerarchica del grafico <i>Year Over Year Change by Most Severe Injury</i> con al di sopra l'elemento di pagina P47_YEARS per il menù a tendina dinamico. b) Selezione della lista dei valori A01_YEARS. c) Visualizzazione del grafico e del menù a tendina subito dopo la selezione del valore '2018'; da notare la comparsa di piccoli indicatori triangolari, rossi e blu, per evidenziare incrementi o decrementi nei valori del grafico	51
 4.1 I tre principali paradigmi di apprendimento di un modello di machine learning	53
4.2 Esempio di algoritmo di classificazione di supervised learning	54
4.3 Esempi di classificazione e di regressione	55
4.4 Rappresentazione grafica della scomposizione di una serie storica nelle sue quattro componenti principali	56
4.5 Comandi eseguiti da terminale per la creazione in locale dell'ambiente virtuale Python e per l'installazione delle librerie necessarie	58
4.6 Script Python per il setup dell'ambiente, la connessione al database e la lettura del dataset	59

4.7	Script Python per l'estrapolazione, la conversione e l'ordinamento dei dati	60
4.8	Script Python per la costruzione della serie storica aggragata settimanalmente	60
4.9	Scelta della finestra di lavoro e suddivisione temporale della serie	61
4.10	Trasformazione della serie in un problema supervisionato con lag feature	62
4.11	Stampa del dataframe contenente il dataset supervisionato	62
4.12	Taglio temporale del dataset supervisionato e creazione delle matrici degli input e dei vettori degli output, da fornire al modello durante il training e il testing	63
4.13	Esempio grafico di un algoritmo Random Forest in un problema di regressione	64
4.14	Creazione e addestramento di un Random Forest Regressor per l'analisi predittiva	64
4.15	Confronto del MAE del modello appena addestrato con quello di un modello base <i>naive</i> , che per ogni settimana prevede lo stesso numero di incidenti della settimana precedente	65
4.16	Grafico di importanza di tutte le feature utilizzate nel modello	66
4.17	Grafico di previsione del numero di incidenti settimanali nel solo intervallo di test	67
4.18	Grafico del test esteso all'intero periodo della serie	68
4.19	Allenamento del modello su tutto il periodo della finestra di lavoro scelta (2016-2023)	68
4.20	Codice per la previsione autoregressiva del modello	69
4.21	Grafico finale dell'analisi predittiva per il 2024 e l'inizio del 2025	70

Introduzione

La sicurezza stradale è un tema di forte rilevanza sociale ed economica. La disponibilità di archivi di grandi dimensioni consente oggi di trasformare i dati in conoscenza utile per orientare decisioni e politiche. In particolare, i dataset sugli incidenti stradali devono descrivere vari aspetti, tra cui il tempo, il luogo, le condizioni, le caratteristiche dell'infrastruttura, i fattori contributivi e la severità degli esiti, così da supportare la prevenzione, la pianificazione della viabilità e le eventuali campagne di sensibilizzazione.

Questa tesi si colloca in tale contesto, prendendo in esame un dataset degli incidenti stradali avvenuti nella città di Chicago e registrati tramite il sistema elettronico E-Crash del Dipartimento di Polizia (CPD) dal 2015 all'inizio del 2024.

A partire da questo archivio di circa 700.000 righe, caratterizzato da molte variabili eterogenee, il lavoro sviluppa un percorso completo che collega l'elaborazione dei dati alla loro analisi e alla successiva visualizzazione tramite grafici dedicati. Più nello specifico, si parte dalla preparazione e dalla modellazione del dataset in Oracle Database, per poi passare alla realizzazione di un applicativo gestionale con delle dashboard tematiche in Oracle APEX, fino a costruire, in un ambiente Python, un primo modello di machine learning per la previsione del numero di incidenti settimanali. L'obiettivo del progetto non è quello di effettuare l'analisi più completa e sofisticata possibile, ma di dimostrare come, partendo da dati grezzi, sia possibile integrare in un unico progetto componenti gestionali, descrittive e predittive, mantenendo un approccio metodico e garantendo la leggibilità e la coerenza delle informazioni presentate.

La tesi è composta da sei capitoli, strutturati nel seguente modo:

- nel Capitolo 1 vengono introdotti il contesto tecnologico e gli strumenti principali utilizzati, con particolare riferimento a Oracle Database, al suo ambiente cloud e al suo strumento APEX di sviluppo web;
- nel Capitolo 2 si descrivono le caratteristiche principali del dataset e la sua elaborazione nel modello relazionale, illustrando le scelte di normalizzazione e documentando la costruzione delle tabelle e delle viste analitiche;
- nel Capitolo 3 si affrontano i concetti di base relativi alla data analytics e alla progettazione dei grafici, illustrando le principali tipologie di analisi e le scelte di visualizzazione adottate per le cinque dashboard tematiche sviluppate in Oracle APEX;
- nel Capitolo 4 si documenta la parte di analisi avanzata, introducendo i concetti fondamentali di machine learning e delle serie storiche, fino ad arrivare alla realizzazione del modello predittivo in Python;

- nel Capitolo 5 si discutono i risultati ottenuti, evidenziando i punti di forza e i limiti del lavoro svolto, sia sul piano applicativo sia su quello metodologico;
- nel Capitolo 6 sono riportate le conclusioni generali e alcuni possibili sviluppi futuri.

CAPITOLO 1

Introduzione ad Oracle APEX

In questo capitolo verrà fatta una panoramica introduttiva su Oracle. In particolare, verrà illustrata l'architettura cloud e il suo Autonomous Database, la cui funzionalità principale è rappresentata dalla piattaforma di sviluppo applicativo low-code denominata APEX. Nella seconda parte del capitolo verranno introdotti tutti gli strumenti APEX principali, con particolare attenzione a quelli utilizzati nel progetto in questione, fino a concludere con un accenno riguardante il caso d'uso dell'argomento preso in analisi.

1.1 Introduzione e contesto

Un settore che attualmente presenta varie criticità è lo sviluppo applicativo in ambito enterprise, ossia la realizzazione di applicativi relativi ad aziende e organizzazioni di medie e grandi dimensioni, come banche, assicurazioni, industrie ed enti pubblici. In queste realtà i sistemi informativi sono complessi, ci sono molti utenti e reparti che devono collaborare e le applicazioni devono essere sicure, integrate con molti altri sistemi e in continuo aggiornamento, in modo da adattarsi in maniera rapida alle mutevoli esigenze aziendali.

Il tradizionale sviluppo di applicazioni per questi contesti complessi presenta vari aspetti negativi. In particolare citiamo:

- *costo elevato*: ogni riga di codice comporta attività di scrittura, debug, manutenzione e aggiornamento, processi che richiedono risorse e personale specializzato costosi;
- *bassa reattività*: i processi rigidi e poco iterativi non riescono ad adattarsi ai continui cambiamenti delle esigenze aziendali;
- *scarsa collaborazione*: sviluppatori, manager e utenti finali operano in maniera isolata, con comunicazione frammentata e inefficace;
- *backlog applicativo*: c'è sempre un accumulo di richieste non soddisfatte e di attività rimaste in sospeso che mantengono le applicazioni non aggiornate, trascurando, in particolare, i problemi minori, ritenuti troppo costosi da affrontare.

Per ridurre questi difetti, negli ultimi anni si sono diffuse le cosiddette piattaforme di sviluppo applicativo “low-code”, cioè degli ambienti di sviluppo rapido che consentono di realizzare applicazioni attraverso interfacce grafiche e componenti predefiniti, riducendo al minimo la scrittura manuale di codice.

Le principali caratteristiche del paradigma low-code possono essere riassunte come segue:

- *rapidità*: l'avvio di un progetto è immediato e lo sviluppo procede con iterazioni veloci;
- *produttività*: anche con sforzo minimo è possibile ottenere applicazioni complete e ricche di funzionalità, focalizzandosi più sui requisiti da rispettare che non sulla corrispettiva scrittura del codice;
- *accessibilità*: lo sviluppo non è riservato ai soli programmatori esperti, ma anche utenti con competenze tecniche di base possono intervenire e contribuire a risolvere problemi aziendali;
- *scalabilità ed estensibilità*: le applicazioni possono nascere con una struttura semplice per poi evolvere in complessità senza eccessivi sforzi, integrando plug-in, requisiti e codice personalizzato quando necessario.

In questo contesto, Oracle APEX rappresenta una delle piattaforme low-code più diffuse a livello enterprise, in grado di realizzare applicativi in maniera rapida e robusta grazie alla stretta integrazione con il database e l'infrastruttura cloud di Oracle.

1.1.1 Che cos'è Oracle APEX

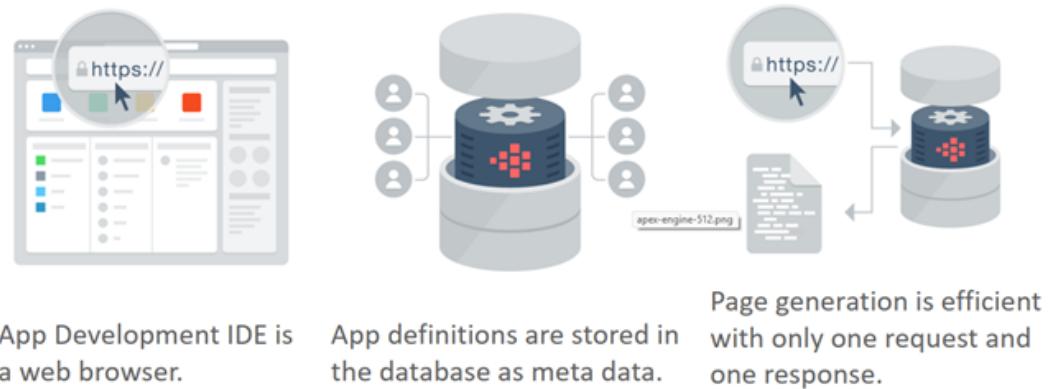
Oracle Application Express (APEX) è la piattaforma di sviluppo low-code di Oracle che consente di creare in maniera rapida e intuitiva applicazioni web aziendali scalabili e con un design che si adatta automaticamente a qualsiasi dispositivo.

APEX è una funzionalità supportata e senza costi aggiuntivi di Oracle Database; questo significa che tutta l'applicazione opera all'interno del database Oracle. In particolare:

- il motore di esecuzione APEX è composto da un insieme di package PL/SQL; perciò tutta la logica riguardante validazioni, flussi applicativi e gestione di utenti e sessioni viene eseguita direttamente nel database;
- le pagine vengono generate a partire da metadati memorizzati nelle tabelle del database, che vengono letti dal motore e trasformati dinamicamente in file HTML, CSS o JavaScript, senza la necessità di file compilati o procedure di rilascio.

Negli applicativi APEX è, quindi, possibile creare rapidamente elementi come report, moduli, grafici e calendari utilizzando i dati salvati nel database e interrogabili tramite SQL. Le principali caratteristiche di Oracle APEX vengono riassunte come segue (Figura 1.1):

- *nessun software client necessario*: l'ambiente di sviluppo è accessibile interamente tramite browser web;
- *sviluppo dichiarativo*: le definizioni delle applicazioni sono memorizzate nel database come metadati, e possono essere aggiornate ed eseguite per visualizzare immediatamente le modifiche, senza la necessità di scrivere codice sorgente;
- *elaborazione dei dati direttamente nel database*: la generazione delle pagine è altamente efficiente, poiché qualsiasi azione che l'utente svolge dal browser, come l'apertura di una pagina o l'invio di una form, viene raccolta in un'unica richiesta al database, il quale restituisce il risultato con una sola risposta.

**Figura 1.1:** Caratteristiche di Oracle APEX

1.2 Stack tecnologico

In questa sezione vengono illustrati l’ecosistema tecnologico su cui si appoggia APEX e le sue principali opzioni di installazione. Verranno introdotti l’Oracle Database come motore dati e di esecuzione, ORDS come punto di accesso web e, infine, ci si soffermerà sull’architettura a tre livelli che riassume l’interazione tra browser, application server e database.

1.2.1 Oracle Database

Oracle Database è un RDBMS, cioè un sistema di gestione di basi di dati relazionale, sviluppato dalla Oracle Corporation, una delle più grandi società tecnologiche che offrono prodotti e servizi per le imprese. Come tutti i database relazionali, Oracle Database consente di gestire i dati secondo uno schema ben strutturato di tabelle collegate tra loro tramite vincoli di integrità referenziale e permette di interrogarli e manipolarli attraverso il linguaggio SQL.

Dal punto di vista della struttura, il database Oracle tiene traccia di tutti i dati e gli oggetti come informazioni registrate nel suo dizionario dati; quest’ultimo consiste in un insieme di tabelle di sistema che fanno da base informativa per la gestione e interrogazione del database. Tra le informazioni principali sono incluse quelle riguardanti tabelle, viste, utenti e permessi.

Tra le varie potenzialità del Database Oracle c’è quella dell’integrazione con PL/SQL, un’estensione procedurale del linguaggio SQL che, insieme a Java, permette di memorizzare ed eseguire procedure, funzioni, trigger e package, al fine di automatizzare operazioni e calcoli sui dati.

Grazie ai suoi vantaggi principali di supporto a grandi quantità di dati, analisi avanzata, backup e sicurezza, Oracle Database è molto utilizzato nelle organizzazioni, soprattutto per la manutenzione di data lake e per il monitoraggio di transazioni in tempo reale.

Tra i casi d’uso principali per Oracle Database vi sono:

- *Applicazioni aziendali e gestione amministrativa:* sviluppo di applicazioni di Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) e di gestione delle risorse umane (HRM), tutti sistemi che devono quotidianamente fornire informazioni e strategie aggiornate.
- *Gestione dei dati e data warehousing:* memorizzazione di vaste quantità di dati strutturati o come data warehouse; quest’ultimi sono sistemi centralizzati che archiviano dati da fonti diverse, per facilitare l’analisi e il reporting.

- *Piattaforme di e-commerce*: gestione dei cataloghi di prodotti, dell'inventario e dei dati delle transazioni.
- *Servizi finanziari e assicurativi*: gestione dei pagamenti e storico delle transazioni, attraverso il sistema sicuro e crittografato, e verifica dei potenziali rischi.
- *Organizzazioni governative*: gestione sicura dei registri e dati pubblici di agenzie governative e di aziende del settore pubblico.
- *Servizi di assistenza sanitaria*: semplificazione delle operazioni, come prenotazioni di visite mediche, e ottimizzazione dei processi.
- *Elaborazione delle transazioni online (OLTP)*: utilizzo della capacità di esecuzione di un gran numero di transazioni di database da parte di un gran numero di utenti in tempo reale, utile, ad esempio, nel commercio elettronico, nelle banche online e nelle aziende di telecomunicazioni.

Dopo aver introdotto il database di Oracle è ora possibile fare un accenno ad Oracle APEX come un ambiente di sviluppo che, oltre a fornire un'interfaccia grafica intuitiva per la creazione di applicazioni, estende il Database Oracle a livello di back-end con una serie di package PL/SQL che aggiungono ulteriori funzionalità.

1.2.2 ORDS

Oracle Rest Data Service (ORDS) è un'applicazione Java che costituisce il punto di accesso web per il database Oracle. In generale ORDS ha diverse funzioni; in particolare:

- offre servizi RESTful per la comunicazione fra applicazioni tramite chiamate web HTTPS e fornisce API utili alla gestione remota del database Oracle;
- abilita SQL Developer Web, un'interfaccia via browser per amministrare e interrogare il database;
- abilita l'ambiente APEX come parte di Database Actions (Figura 1.2), un'interfaccia web che fornisce funzionalità di sviluppo, strumenti di gestione dati, amministrazione e monitoraggio per Oracle Database.

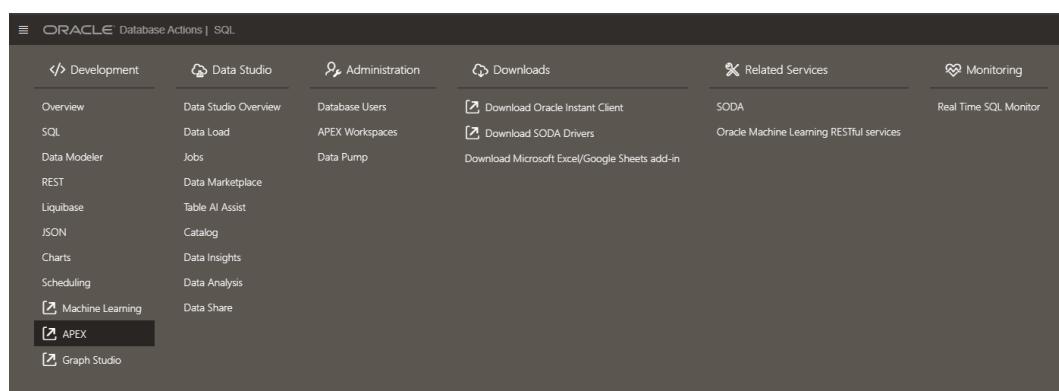


Figura 1.2: Elenco delle funzionalità di Database Actions offerte da ORDS

Nell'ambito di questo progetto, ORDS svolge la funzione principale di abilitazione di un server web per l'applicativo APEX, il quale gestisce ogni richiesta del browser, come il

rendering delle pagine o l'invio di una form, la inoltra al database Oracle e restituisce al client la risposta, tutto tramite chiamate web HTTPS. Questo processo semplifica il funzionamento dell'applicazione e garantisce la sicurezza durante lo scambio dei dati.

1.2.3 Architettura a tre livelli

L'ambiente di sviluppo APEX si basa, quindi, su un'architettura a tre livelli (Figura 1.3):

1. *Browser* (web client): applicativo APEX dal quale l'utente manda le richieste;
2. *ORDS* (application server): gestore delle richieste che vengono inoltrate al database;
3. *Database Oracle con APEX*: motore di esecuzione che processa le richieste e restituisce al browser le risposte tramite ORDS.

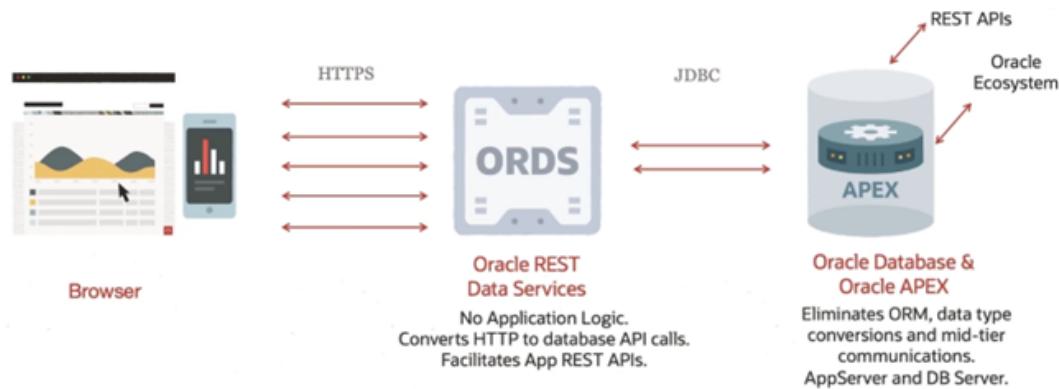


Figura 1.3: Architettura a tre livelli di Oracle APEX

Durante la creazione o l'estensione di un'applicazione, APEX genera o aggiorna i metadati memorizzati nelle tabelle del database. Quando l'applicazione viene eseguita, il motore APEX legge i metadati e genera dinamicamente la pagina richiesta o elabora i dati inseriti dall'utente. Non è necessaria alcuna compilazione basata su file né alcuna generazione di codice; tutte le operazioni sono gestite direttamente tramite SQL sugli schemi dati del database.

È possibile installare in completa autonomia tutto lo stack tecnologico illustrato finora secondo la seguente cronologia:

1. installazione del DBMS Oracle su un server;
2. installazione dei pacchetti APEX nel DBMS;
3. installazione di ORDS su di un server, che può anche coincidere con quello in cui si è installato il DBMS Oracle.

1.2.4 Opzioni di installazione

Sono stati discussi finora tutti i componenti necessari allo stack tecnologico, dal DBMS Oracle con i pacchetti APEX all'application server di ORDS, ma dove può essere installata tutta questa struttura?

L'installazione dello stack può avvenire su diversi ambienti (Figura 1.4), ovvero:

- *in locale*: su un computer personale, utilizzando una versione ridotta e più leggera del DBMS Oracle, chiamata Oracle Free 23ai;

- *on-premise*: su un server aziendale, scegliendo tra l'edizione Standard (SE) o quella Enterprise (EE);
- *in cloud*: su un opportuno server generato come virtual machine;
- *in un Cloud Oracle (OCI)* attraverso il servizio in PaaS (Platform as a Service) *Autonomous Database*, un DBMS che Oracle installa e fornisce già pronto per l'utilizzo.

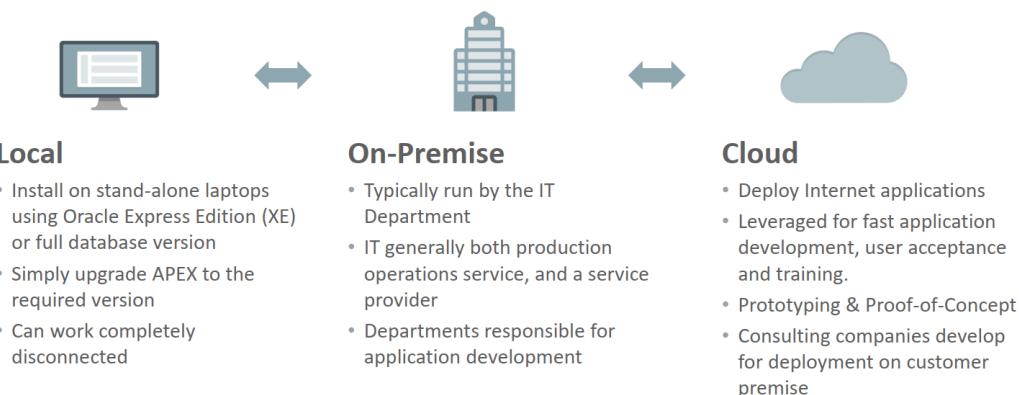


Figura 1.4: Opzioni di installazione dello stack tecnologico

Come verrà illustrato nella prossima sezione, nel caso di questo progetto è stato utilizzato un Autonomous Database sul Cloud Oracle, dopo aver richiesto un piano gratuito per accedere ai servizi *Always Free*.

1.3 OCI e Autonomous Database

In questa sezione viene illustrata l'infrastruttura del Cloud di Oracle (OCI) e tutti i servizi che offre, tra cui Autonomous Database e Oracle APEX.

1.3.1 Oracle Cloud Infrastructure

Oracle Cloud Infrastructure (OCI) è la piattaforma cloud di Oracle che offre un set di servizi per lo sviluppo e l'esecuzione di una vasta gamma di applicazioni in un ambiente hosted ad alta disponibilità. OCI fornisce funzionalità di computazione a elevate prestazioni, come istanze hardware fisiche, e capacità di storage in una rete virtuale flessibile e accessibile in modo sicuro.

La struttura della piattaforma Cloud di Oracle può essere riassunta come segue, partendo dall'architettura fisica fino alla descrizione dei servizi offerti (Figura 1.5):

- *Regioni cloud*: aree e domini di disponibilità in cui si trovano i server fisici comprati da Oracle per ospitare tutta l'infrastruttura cloud.
- *Core dell'infrastruttura*; esso consiste di:
 - server fisici o singole macchine virtuali;
 - container, come Docker o Kubernetes;
 - VMWare, ossia dei software per la creazione di server virtuali;
 - storage;
 - networking, ossia creazione e gestione di reti private o pubbliche.

- *Database:*
 - insieme dei DBMS Oracle, riferiti principalmente ad Autonomous Database, tutti operanti come Platforms as a Service (PaaS), cioè come servizi dei quali non bisogna manualmente fare l'installazione e il setup, ma che vengono direttamente forniti da Oracle già pronti all'uso;
 - altri DBMS, come MySQL e PostgreSQL.
- *Gestione e amministrazione:* monitoring, logging, sicurezza e gestione di tutte le operazioni.
- *Strumenti per sviluppatori,* tra cui Oracle APEX.

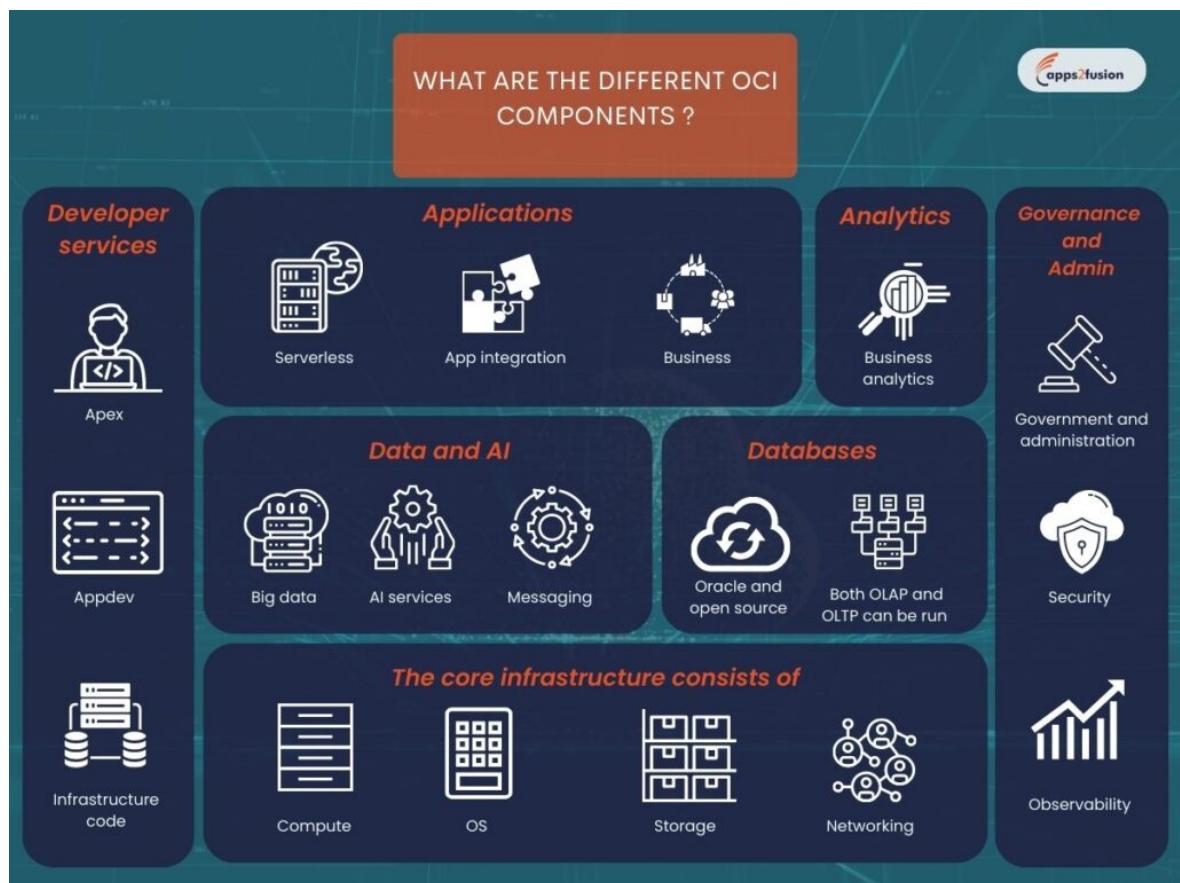


Figura 1.5: Infrastruttura completa del cloud di Oracle

Nell'ambito del lavoro svolto per questa tesi è stato utilizzato il servizio OCI dell'Autonomous Database, il quale ha effettuato il setup automatico di un DBMS con integrato l'ambiente di sviluppo APEX.

A tal proposito è stato richiesto un Oracle Cloud Free Tier (Figura 1.6), un piano gratuito che consente di creare un account Oracle Cloud munito di una serie di servizi, sempre gratuiti, disponibili per un periodo di tempo illimitato, tra cui oltre ad APEX: storage di oggetti e archivi, Autonomous Data Warehouse e Transaction Processing e monitoraggio delle performance delle applicazioni.

The screenshot shows the Oracle Cloud Free Tier services page and a summary card for APEX on Autonomous Database.

Services available on Oracle Cloud Free Tier:

- Search bar: Search
- Filtering: Always Free X, Oracle Databases X
- Featured Product dropdown: Al and Machine Learning, Compute, Networking, Oracle Databases, Storage
- Tier Type dropdown: Always Free (selected), Trial
- Result: **ALWAYS FREE** Oracle Databases **Autonomous Database**. Description: Oracle Autonomous Transaction Processing, Autonomous Data Warehouse, Autonomous JSON Database, or APEX Application Development. Capacity: Up to Two databases total.

APEX su Oracle Cloud:

- Description: È possibile sviluppare e distribuire applicazioni low-code in un ambiente APEX sicuro e completamente gestito sulla prima e unica infrastruttura Autonomous al mondo.
- Buttons: Visualizza i prezzi, Guarda i video
- Benefits (checkmarks):
 - Due ambienti Sempre gratis, ognuno con ECPU e circa 20 GB di memoria inclusi, che supportano l'upgrade con un clic per sbloccare più risorse
 - Nessun limite per le app, gli utenti o gli sviluppatori
 - Monitoraggio, backup, applicazione di patch e upgrade automatici
 - Altamente disponibile con scala automatica
 - Disponibile in più di 40 aree cloud pubbliche
 - Conforme alle normative regionali sui dati, incluse FedRAMP, HIPAA, SOC 1-3, PCI

Figura 1.6: Servizi dell’Oracle Cloud Free Tier e APEX su Autonomous Database

1.3.2 Autonomous Database

Oracle Autonomous Database è un DBMS in cui l’installazione e il setup sono completamente gestiti da Oracle; l’infrastruttura cloud mette a disposizione un server per poi procedere con le installazioni di tutti i componenti necessari, in modo da fornire allo sviluppatore un DBMS Oracle pronto all’uso.

Autonomous Database offre diverse modalità d’uso rispetto alle quali è possibile ottimizzare il DBMS fornito:

- *Autonomous Transaction Processing* (ATP): ottimizzato per operazioni transazionali, come quelle di un gestionale o un’app web con molti utenti che inseriscono e aggiornano dati contemporaneamente.
- *Autonomous Data Warehouse* (ADW): ottimizzato per analisi e reporting su grandi volumi di dati.
- *Autonomous JSON Database* (AJD): ottimizzato per applicazioni che operano con dati in formato JSON.

Tra tutti gli strumenti forniti dall’Autonomous Database, due principali sono proprio l’ambiente di sviluppo APEX, che verrà trattato nella sezione successiva, e Oracle Machine Learning, per il supporto all’analisi dei dati e alla creazione di modelli di machine learning.

1.3.3 APEX con Autonomous Database

Come appena accennato, Oracle APEX è una funzionalità del DBMS Oracle; tale funzionalità mette a disposizione un ambiente di sviluppo applicativo low-code.

Alcuni dei vantaggi principali di APEX sono elencati di seguito:

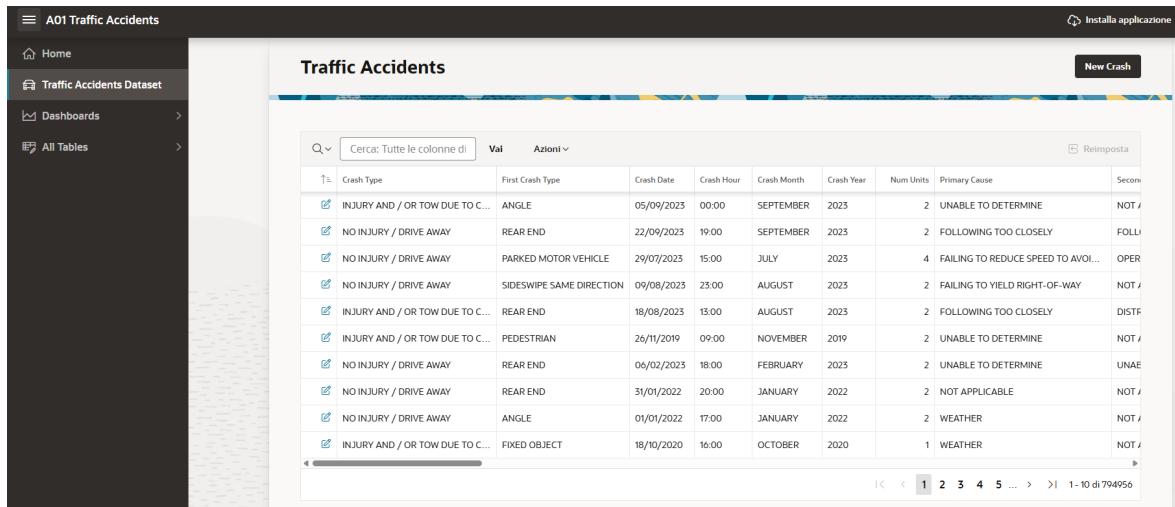
- *interfaccia utente responsive*, capace quindi di adattarsi a più dispositivi;

- *grandi capacità di elaborazione e visualizzazione dei dati:* APEX presenta funzionalità per il reporting e la creazione di grafici, oltre a strumenti per la manipolazione dei dati;
- *supporto automatico alla lingua e alla localizzazione;*
- *monitoraggio e logging,* per tenere traccia del flusso di utenti e delle loro azioni all'interno delle applicazioni.

Tra principali casi d'uso si evidenziano:

- *Sostituzione di fogli di calcolo:* l'ambiente APEX permette di importare al suo interno file XLS, CSV, XML o JSON per poi convertirli automaticamente in tabelle da usare come base per costruire un applicativo. Ciò permette anche di migliorare la gestione e la visualizzazione di tali dati; quello che prima era un foglio Excel può, ad esempio, diventare una griglia interattiva (Figura 1.7), molto più semplice e rapida da gestire e da mostrare all'utente finale, anche semplicemente condividendo l'URL dell'applicativo.
- *Sviluppo ed estensione di sistemi aziendali complessi:* APEX aiuta nella creazione e nella scalabilità di ambienti che devono gestire grandi quantità di dati archiviati in sistemi diversi e che necessitano la realizzazione di report quanto più dettagliati possibili, limitando l'accesso a informazioni sensibili e gestendo le autorizzazioni degli utenti.

Riguardo al progetto in questione, si illustrerà più avanti come, tramite la piattaforma Oracle APEX, è stato realizzato un applicativo di tipo gestionale e di reporting per la gestione e la visualizzazione di dati su incidenti stradali forniti dal Dipartimento di polizia della città di Chicago.



The screenshot shows the 'Traffic Accidents' page of the A01 Traffic Accidents application. The left sidebar contains navigation links: Home, Traffic Accidents Dataset, Dashboards, and All Tables. The main area has a title 'Traffic Accidents' and a search bar. Below is a table with columns: Crash Type, First Crash Type, Crash Date, Crash Hour, Crash Month, Crash Year, Num Units, Primary Cause, and Secondary Cause. The table lists various accident types like INJURY AND / OR TOW DUE TO C..., NO INJURY / DRIVE AWAY, etc., with details such as date (e.g., 05/09/2023), time (e.g., 00:00), month (e.g., SEPTEMBER), year (e.g., 2023), number of units (e.g., 2), primary cause (e.g., UNABLE TO DETERMINE), and secondary cause (e.g., NOT APPLICABLE). At the bottom, there are navigation buttons for pages 1 through 5, and a total count of 1-10 di 704956.

Figura 1.7: Pagina dell'applicativo contenente una griglia interattiva con tutti i record degli incidenti stradali

1.4 Ambiente e strumenti di Oracle APEX

Appena effettuato l'accesso ad un account APEX ci si trova subito nella pagina principale del workspace, ovvero l'ambiente di lavoro messo a disposizione, dal quale è possibile direttamente iniziare ad esplorare gli oggetti del database e realizzare applicativi.

Dalla home page del workspace (Figura 1.8) è possibile accedere agli strumenti chiave di sviluppo; questi sono:

1. *App Builder*: strumenti per la creazione di un'applicazione basata sugli oggetti del database;
2. *SQL Workshop*: strumenti di accesso al database per la gestione e la visualizzazione di tutti i suoi oggetti;
3. *Team Development*: strumenti di aiuto per team di sviluppatori nella gestione e nel monitoraggio di eventuali problemi nelle applicazioni;
4. *Gallery*: set di applicativi d'esempio già sviluppati, dai quale prendere spunto o partire direttamente come base da estendere e personalizzare.

Ciascuno di questi strumenti appare nel menù di navigazione collocato in alto e mostra un elenco a discesa con tutti i relativi componenti e le relative funzioni.

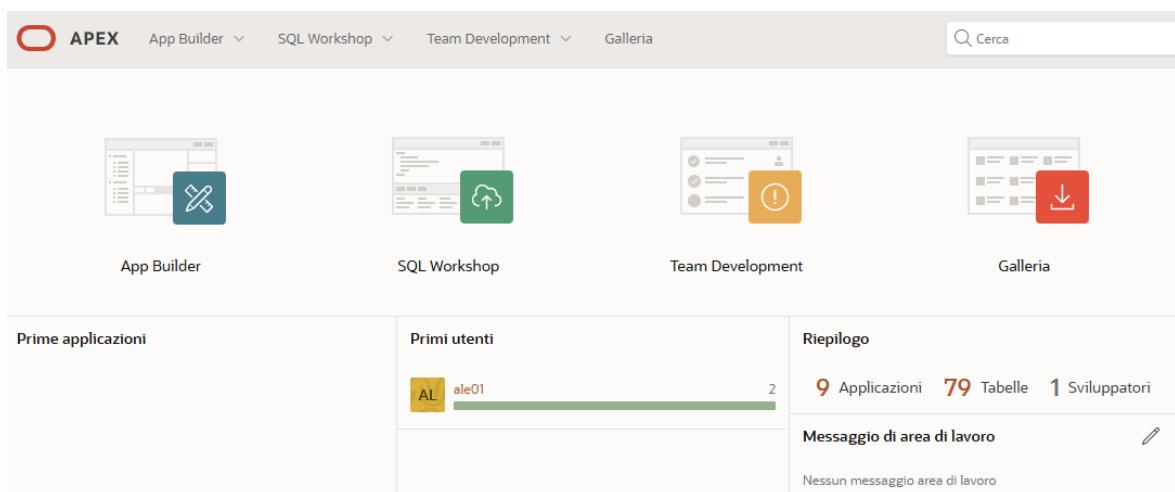


Figura 1.8: Home page dell'area di lavoro APEX

1.4.1 Workspace

In Oracle APEX un workspace è un ambiente di lavoro isolato che consente a più utenti di operare sulla stessa installazione di APEX mantenendo separati e privati i propri oggetti, i propri dati e le proprie applicazioni. Può essere considerato come una sorta di database virtuale privato, identificato da un ID numerico univoco e da un nome. Ogni istanza dall'ambiente di sviluppo APEX può contenere più workspace, ciascuno dedicato ad applicazioni e gruppi di lavoro differenti.

1.4.2 SQL Workshop

Tra gli strumenti chiave della piattaforma APEX, SQL Workshop è quello che consente agli sviluppatori di gestire direttamente dal browser tutti gli oggetti del database.

Come prima cosa, nella pagina dedicata a questa sezione è possibile visualizzare a destra l'elenco degli schemi di database dal quale selezionare quello predefinito da utilizzare. Con schema si intende il contenitore logico che raccoglie tutti gli oggetti del database tra cui: tabelle, viste, indici, procedure, funzioni e pacchetti PL/SQL. In APEX ogni workspace è associato a uno o più schemi; alcuni di questi sono di sistema e non possono essere utilizzati poiché riservati; gli altri, autorizzati dall'amministratore del database, possono essere assegnati ai workspace. In particolare, è detto parsing schema quello scelto come sorgente di dati per le applicazioni, che in questo caso è WKSP_ALE01.

In base alle operazioni da compiere, la sezione SQL Workshop (Figura 1.9) offre cinque componenti:

1. *Browser Oggetti*: consente di esplorare, creare e modificare oggetti del database attraverso un’interfaccia grafica intuitiva.
2. *Comandi SQL*: comprende un editor per scrivere istruzioni SQL o blocchi di codice PL/SQL al fine di creare, modificare, visualizzare ed eliminare oggetti del database.
3. *Script SQL*: sezione utilizzata per creare e salvare come file degli script SQL, contenenti, ad esempio, operazioni di creazione o modifica di tabelle e viste.
4. *Utility*: comprende tutte le funzionalità APEX per creare query SQL, caricare e scaricare dati da un database Oracle, generare istruzioni DDL, gestire le impostazioni predefinite dell’interfaccia utente, monitorare il database e visualizzarne i dettagli.
5. *Servizi RESTful*: consente di creare e gestire servizi RESTful utilizzando la libreria di servizi REST di Oracle, ovvero Oracle REST Data Services (ORDS).

Al di fuori dei servizi RESTful, in questo progetto sono stati utilizzati tutti i componenti di SQL Workshop, rispettivamente per caricare i dati del dataset nella piattaforma, per creare le tabelle e le viste necessarie, sia tramite script SQL sia tramite interfaccia grafica, e infine per realizzare e testare tutte le istruzioni SQL necessarie alla realizzazione dei grafici delle dashboard.

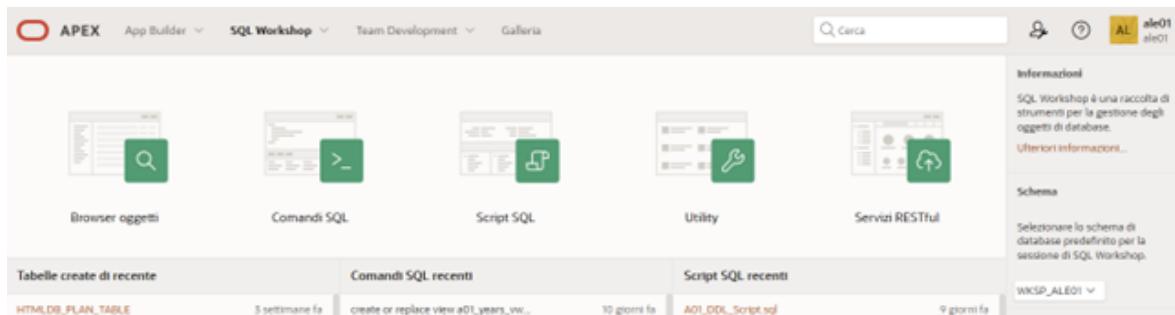


Figura 1.9: Pagina di SQL Workshop con i suoi strumenti

In Browser Oggetti vengono visualizzate le proprietà degli oggetti presenti nel database, con la possibilità di crearne di nuovi.

La pagina dedicata è suddivisa in due riquadri (Figura 1.10):

1. *riquadro di selezione degli oggetti*: appare sul lato sinistro ed elenca tutti gli oggetti del database all’interno dello schema corrente;
2. *riquadro dei dettagli*: appare sul lato destro e visualizza informazioni di dettaglio sull’oggetto selezionato.

Il riquadro dei dettagli presenta varie sezioni; in particolare Colonne, Dati e DDL sono state quelle più utilizzate per visualizzare la struttura delle tabelle, i loro record e la query di creazione generata in automatico attraverso la procedura grafica guidata.

Figura 1.10: Visualizzazione della tabella A01_TRAFFIC_ACCIDENTS nel Browser Oggetti

In Comandi SQL è disponibile un editor testuale nel quale digitare ed eseguire istruzioni SQL (Figura 1.11).

Questa sezione è stata molto sfruttata nel corso del progetto per creare e testare tutte le query SQL per il recupero dei dati necessari alla realizzazione dei grafici. Nella parte sotto-stante l'editor è presente una finestra dal contenuto dinamico, che comprende il terminale per i risultati delle query e la cronologia di tutte le istruzioni eseguite fino a quel momento.

In Script SQL sono raccolti tutti gli script realizzati per creare, modificare, visualizzare, eseguire ed eliminare oggetti del database. Con script SQL si intende un set di comandi SQL, salvati come file, che può contenere una o più istruzioni SQL o blocchi PL/SQL.

WEATHER_CONDITION	CRASH_COUNT
CLEAR	624964
RAIN	69694
UNKNOWN	43295
SNOW	26895

Figura 1.11: Scrittura ed esecuzione della query nell'editor di Comandi SQL per recuperare i dati relativi alle condizioni meteo degli incidenti stradali

In particolare, questa sezione dell'SQL Workshop viene sfruttata per:

- *gestire gli script*: si possono creare nuovi script tramite editor o caricare file esistenti;
- *visualizzare i risultati*: quando si esegue uno script viene visualizzata una pagina di risultati con l'esecuzione di ogni istruzione e il feedback corrispondente;
- *creare un'applicazione*: cliccando sul pulsante 'Crea applicazione' all'interno dell'editor viene analizzato lo script e viene creata direttamente un'applicazione.

L'applicativo sviluppato per la gestione del dataset sugli incidenti stradali si basa interamente sulle tabelle e sulle viste realizzate tramite questi script. In particolare, qui si trovano gli script SQL per la creazione e l'aggiornamento di tutte le tabelle di lookup, e soprattutto della vista `a01_traffic_accidents_vw` contenente i dati ripuliti e raggruppati.

In Utility è possibile utilizzare la sezione Data Workshop per caricare nell'area di lavoro dati caratterizzati da vari tipi di formati di file, quali XLSX, CSV, XML e JSON. È, inoltre, possibile utilizzare la procedura guidata 'Scarica dati' per esportare i dati in formato testo o XML.

Il dataset degli incidenti stradali, argomento di questo progetto, è stato scaricato dal sito di Kaggle in formato CSV e caricato nel workspace APEX attraverso la procedura 'Carica dati', la quale verrà illustrata più in dettaglio nel capitolo seguente.

1.4.3 App Builder

Analogamente alla struttura di SQL Workshop, la sezione dell'App Builder, dedicata alla creazione, modifica e monitoraggio di un'applicazione, è suddivisa in vari componenti in base alle sue funzioni principali (Figura 1.12). Queste ultime sono:

- *Creazione di un'applicazione*: cliccando sulla sezione 'Crea' compaiono una serie di opzioni, quali la creazione guidata tramite interfaccia grafica, la creazione da un file CSV, XLSX, XML o JSON, la creazione da Quick SQL o l'opzione di utilizzo di una delle applicazioni già pronte nella Galleria.
- *Importazione ed esportazione*: è possibile esportare o importare le applicazioni o i loro backup sotto forma di file SQL non codificati, in modo da spostarli con facilità da un ambiente all'altro.
- *Monitoraggio delle applicazioni*: la sezione 'Dashboard' presenta grafici e metriche relative alle applicazioni dell'area di lavoro, fornendo informazioni come la media di pagine per ogni app e statistiche sulle pagine più attive e i componenti più utilizzati al loro interno.

In Oracle APEX un'applicazione è definita principalmente da pagine e componenti condivisi. Con quest'ultimo termine si intende un set di strumenti utilizzati per definire vari elementi comuni che possono essere utilizzati più volte nell'ambito di una certa applicazione. Alcuni esempi sono: il menu di navigazione, gli attributi dell'interfaccia utente, gli schemi di autenticazione e le liste dei valori per gli elenchi di selezione.

Cliccando su una delle applicazioni già sviluppate si apre una nuova sezione dell'App Builder ad essa relativa. Qui sono presenti funzioni aggiuntive come 'Crea pagina', per la creazione guidata di nuove pagine, o 'Page Designer', per la modifica di pagine esistenti.

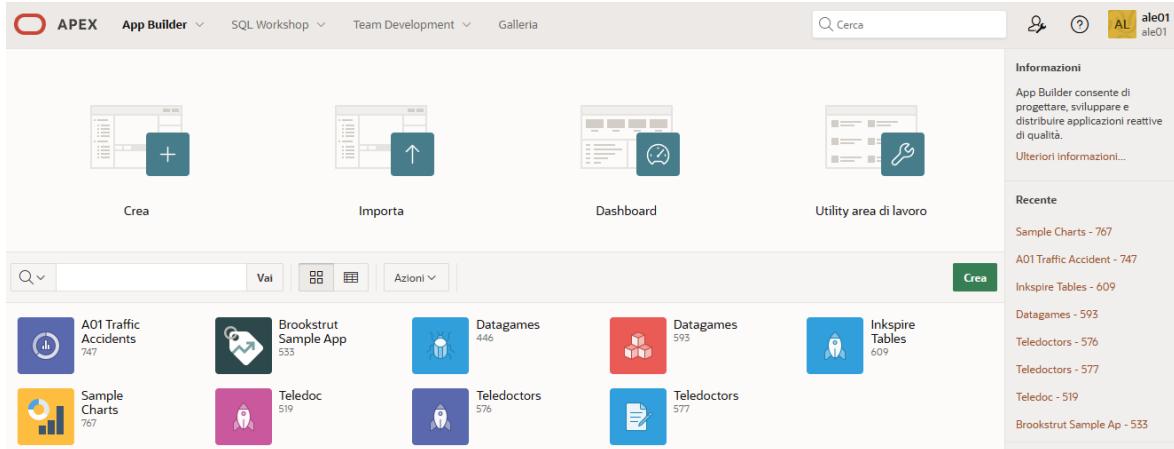


Figura 1.12: Pagina dell’App Builder con tutti i suoi strumenti

Cliccando sul pulsante ‘Crea pagina’ compare una modale (Figura 1.13) con tutte opzioni da poter realizzare: da pagine vuote da popolare manualmente, fino a pagine più complesse contenenti report interattivi, calendari e dashboard già pronti, basati sugli oggetti del database ad essi collegati.

Il Page Designer offre un ambiente che include una barra degli strumenti e numerosi riquadri per la modifica delle pagine.

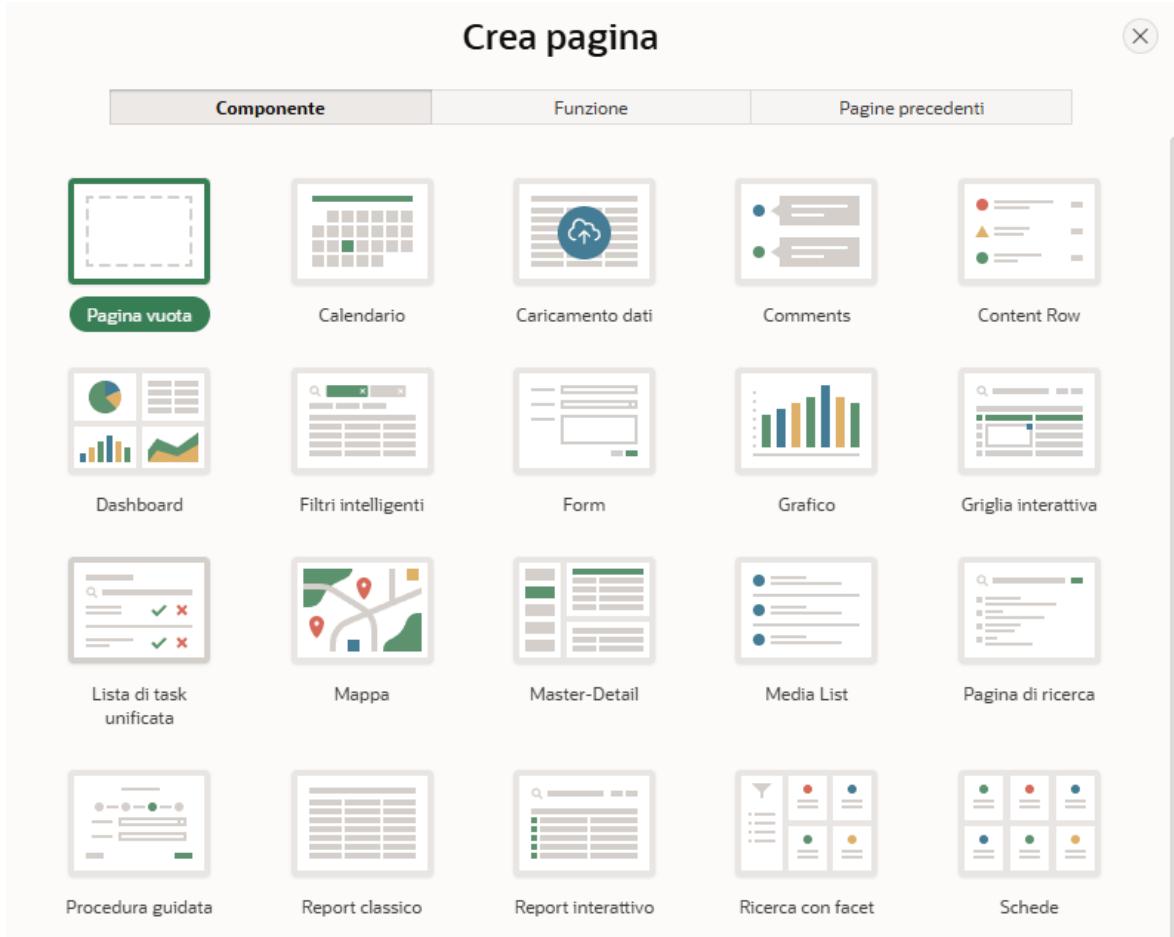


Figura 1.13: Modale di selezione della tipologia di pagina da creare

Esso è suddiviso in tre riquadri principali (Figura 1.14), ovvero:

1. *Componenti di pagina*: riquadro di sinistra che include il rendering, le azioni dinamiche e i componenti condivisi della pagina. In genere in questa sezione si interagisce con la struttura di rendering della pagina per selezionare rapidamente uno o più componenti su cui lavorare nell'editor delle proprietà situato nel riquadro di destra.
2. *Layout*: riquadro centrale che include la scheda layout della pagina, contenente una vista astratta in cui è possibile creare e ridisporre tramite trascinamento i componenti di pagina.
3. *Editor delle proprietà*: riquadro di destra che consente di visualizzare e modificare contemporaneamente tutte le proprietà relative a uno o più componenti della pagina. Proprietà simili vengono raggruppate in sezioni comprimibili. Nella parte superiore della schermata è presente un filtro per la ricerca.

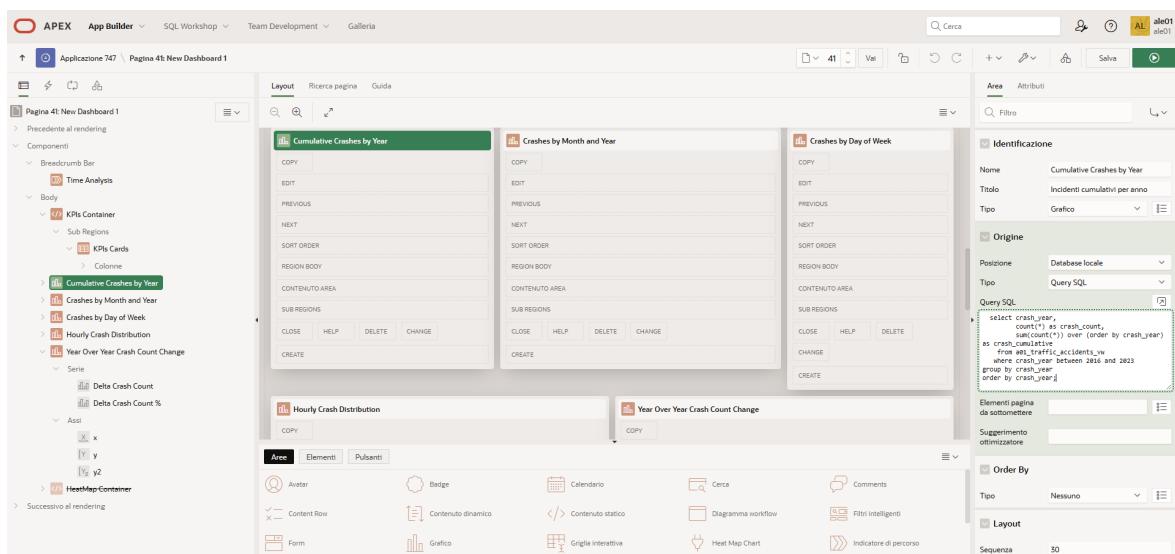


Figura 1.14: Page Designer che mostra la struttura e l'editor di modifica della pagina numero 41 dell'applicazione

1.5 Collegamento al caso di studio

Dopo aver introdotto tutto l'ambiente Oracle APEX e illustrato brevemente il processo di sviluppo di un'applicazione, ci si può spostare al caso di studio del progetto, che riguarda un dataset di incidenti stradali avvenuti nella città di Chicago tra il 2015 e l'inizio del 2024.

1.5.1 Gestionale e reporting

Il processo, che verrà illustrato in dettaglio nel capitolo seguente, riguarda la creazione di un'applicazione basata sul dataset Kaggle importato su Oracle APEX e trasformato in tabelle e viste con dati rielaborati. Lo sviluppo applicativo è stato fatto tenendo a mente due principali obiettivi:

- *Gestione del dataset*: per ogni tabella si ha un report o una griglia interattiva con form (Figura 1.15), in modo da poter effettuare, in maniera semplice e comoda, operazioni di

aggiunta, modifica ed eliminazione dei record. In breve, queste sezioni dell'applicativo contenenti i report svolgono la stessa funzione di un registro digitale per salvare gli incidenti stradali e tutte le categorie dei campi da compilare, riguardanti, ad esempio, la tipologia di incidente, le informazioni temporali e geografiche e quelle su eventuali danni e feriti.

- **Visualizzazione dati e reporting:** partendo da query SQL per recuperare e analizzare i dati delle tabelle, si possono costruire innumerevoli grafici filtrati e interattivi di diverse tipologie (Figura 1.16). In questo caso sono state realizzate cinque dashboard tematiche per rappresentare visivamente diverse analisi e statistiche sugli incidenti stradali.

Figura 1.15: Griglia interattiva con form per l'inserimento e la modifica dei record della vista sugli incidenti stradali

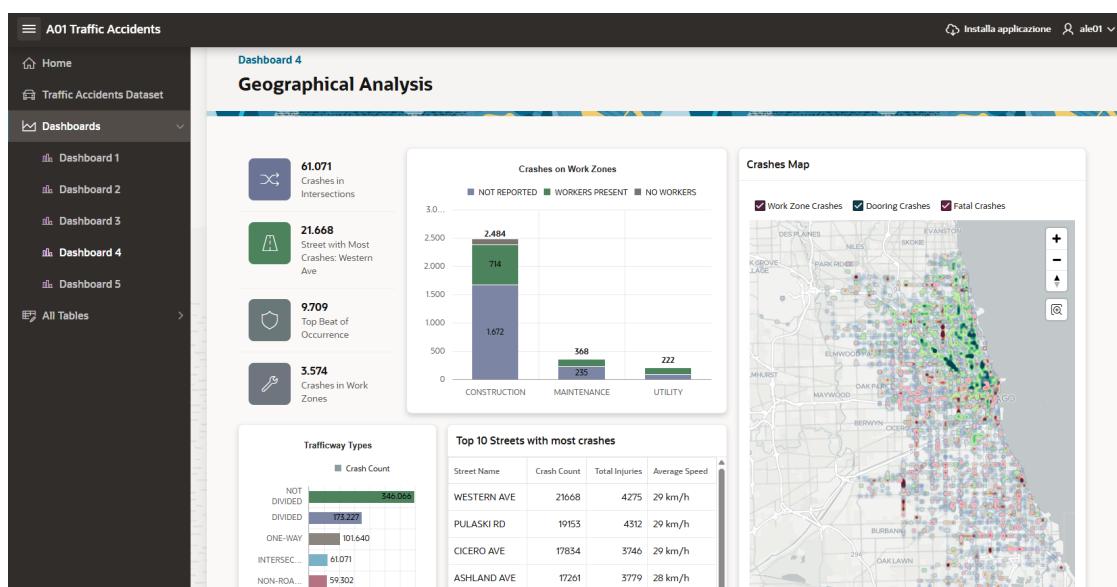


Figura 1.16: Esempio di dashboard tematica con KPI e grafici interattivi

CAPITOLO 2

Descrizione dei dati di partenza e attività di pre-processing

In questo capitolo verranno illustrati i dati di partenza utilizzati per il progetto e tutte le principali attività di pre-processing necessarie alla loro analisi e visualizzazione nell'applicativo APEX. Nella prima parte si parlerà del dataset selezionato, della sua struttura e delle ragioni che hanno portato alla sua scelta. Successivamente, verranno documentati i metodi di elaborazione dei dati, la normalizzazione del database e le operazioni di pulizia e aggregazione realizzate attraverso una vista. La seconda parte del capitolo sarà, invece, dedicata alla descrizione dell'applicativo gestionale realizzato in APEX per visualizzare e gestire i dati attraverso una griglia e dei report interattivi.

2.1 Dataset

Con il termine "dataset" vengono generalmente indicati gruppi di dati correlati, che condividono gli stessi attributi e proprietà, e che possono presentarsi in formati differenti: collezione di immagini in una cartella, file CSV, file XML o modelli con una struttura definita, come i database. In questa sezione si analizzeranno le caratteristiche del dataset preso come riferimento e le motivazioni che hanno portato a sceglierlo.

2.1.1 Scelta e struttura del dataset

La scelta del dataset di riferimento è molto importante e si basa sulla tipologia di analisi che si intendono effettuare. Nel caso di questo progetto è stato preso, dal sito Kaggle, un dataset contenente informazioni riguardo agli incidenti stradali nella città di Chicago, registrate dal sistema elettronico di segnalazione degli incidenti (E-Crash) del Dipartimento di Polizia (CPD).

Questo dataset (Figura 2.1) comprende circa 50 colonne caratterizzate da categorie e tipi di dato differenti. Ogni record rappresenta un singolo incidente stradale e include molti dettagli, come data, ora, localizzazione, condizioni e fattori contributivi.

La scelta è stata fatta tenendo in considerazione i vantaggi principali di tale dataset. Quest'ultimo, infatti, è:

- *di facile comprensione e ben documentato;*
- *di dimensioni moderate:* occupa massimo 450 MB di memoria;
- *ragionevolmente pulito:* non contiene troppi errori, duplicati o valori mancanti;

- con licenza aperta: utilizza Apache License 2.0, una licenza open source che ne permette l'uso, la modifica e la distribuzione, anche a scopo commerciale.

Grazie alle caratteristiche appena elencate, il dataset scelto risulta versatile per esplorare schemi ricorrenti, fattori contributivi e condizioni legate a ciascun incidente, fornendo eventuali informazioni preziose per la sicurezza stradale e per la pianificazione urbana.

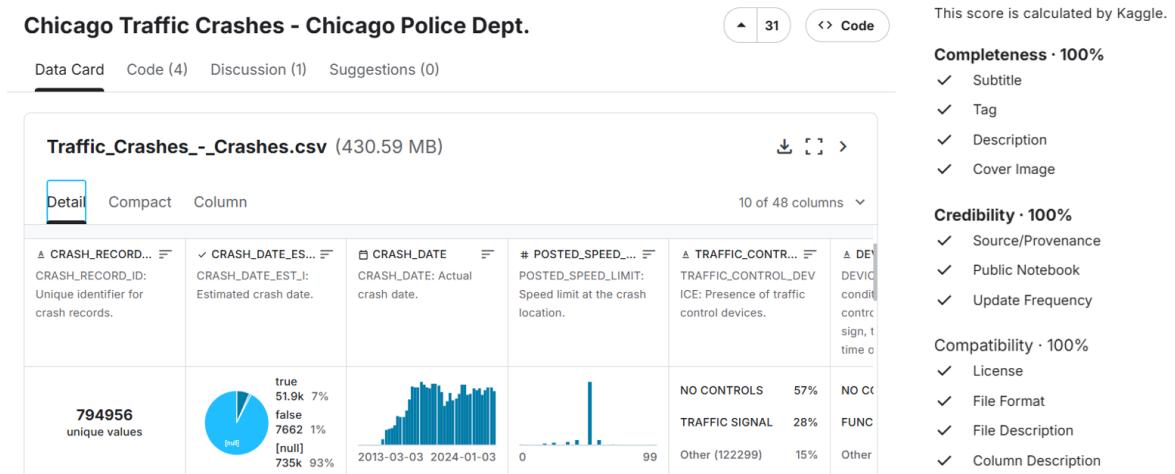


Figura 2.1: Struttura e valutazione del dataset preso dal sito Kaggle

2.2 Metodi di elaborazione dei dati

Con "elaborazione dei dati" si definisce il processo di raccolta dei dati e la loro trasformazione in informazioni fruibili. Questo processo inizia a partire da dati grezzi, i quali vengono estratti da varie sorgenti e in seguito convertiti in un formato più facilmente leggibile. Dopo aver assunto la forma e il contesto necessari per consentire l'interpretazione e l'impiego, i dati possono essere utilizzati in diverse analisi per rispondere ad esigenze specifiche, come, ad esempio, la previsione dell'esito di campagne decisionali e la generazione di report e dashboard.

Due tra i principali modelli di elaborazione dei dati sono l'ETL e l'ELT. Di seguito verranno prima esaminate alcune analogie e differenze tra questi due approcci e poi si spiegheranno la scelta e l'implementazione fatte in questo progetto.

La principale differenza tra ELT ed ETL sta nell'ordine in cui vengono eseguite le operazioni. *ETL* è l'acronimo di "Extract, Transform and Load". Questo processo prevede, come primo passo, l'estrazione dei dati dalla fonte, seguita dalla loro trasformazione all'interno di un'area di staging, e infine dal loro trasferimento in un archivio centralizzato, dove saranno accessibili per l'analisi (Figura 2.2). Con il termine "area di staging" si fa riferimento a un'area di archiviazione intermedia per salvare in via temporanea i dati estratti. Le aree di staging dei dati sono spesso transitorie e i loro contenuti vengono solitamente cancellati una volta completata l'estrazione.

L'ETL è stato, per alcuni decenni, il modello standard utilizzato nell'elaborazione dei dati, mentre l'*ELT* è un'opzione più recente che sfrutta le moderne capacità di archiviazione dei dati. *ELT* è l'acronimo di "Extract, Load and Transform", il che significa che il caricamento dei dati in un archivio centralizzato avviene subito dopo la loro estrazione. In questo caso la trasformazione dei dati in un formato utilizzabile avviene solo in seguito, direttamente dall'archivio e nel momento in cui se ne ha necessità (Figura 2.3).

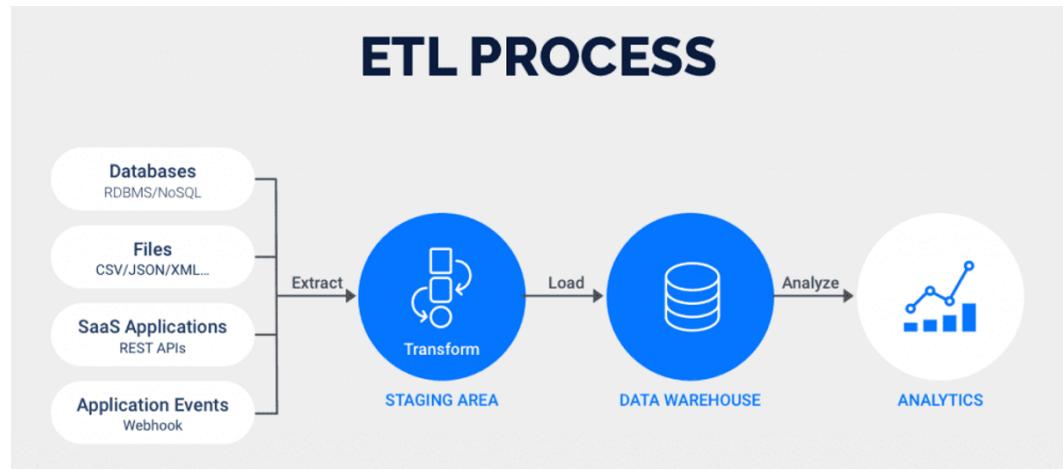


Figura 2.2: Processo di ETL

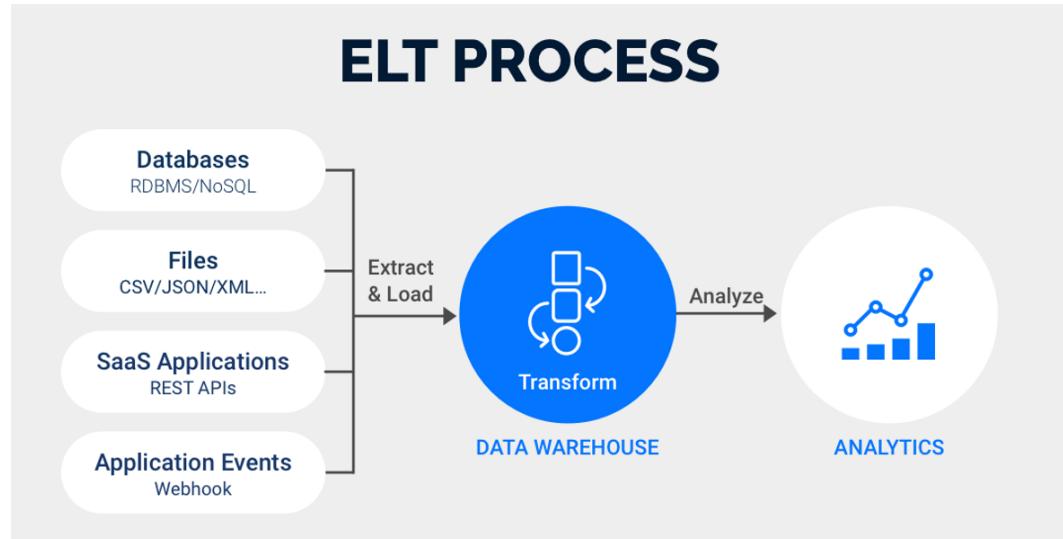


Figura 2.3: Processo di ELT

Nel caso del progetto corrente è stato utilizzato un approccio di tipo ELT. Come prima cosa, il dataset è stato estratto dal sito di riferimento, poi è stato importato nel DBMS Oracle e, infine, è stato normalizzato e rielaborato al bisogno per l'analisi e la visualizzazione tramite report e grafici.

2.2.1 Estrazione e caricamento

Dopo l'estrazione in formato CSV dal sito di Kaggle, il dataset viene caricato nel DBMS Oracle tramite la procedura di *data loading* di APEX. Per attuare quest'ultima, si parte dall'area dell'SQL Workshop, si naviga nella sezione 'Utility' e si sceglie 'Data Workshop'. Da qui è possibile eseguire procedure di esportazione e importazione dei dati in diversi formati, compreso il CSV.

Con l'opzione 'Carica dati' si importa il file `Kaggle_traffic_accidents_dataset.csv`. La procedura permette di effettuare un parsing automatico di tutte le righe e colonne del dataset in un'unica tabella che farà da archivio centrale. Durante questo processo si sceglie il nome della tabella che, in questo caso, è stata rinominata `A01_TRAFFIC_ACCIDENTS`, e si

selezionano i tipi di dati, con le relative lunghezze, di tutte le colonne campionate (Figura 2.4).

The screenshot shows a user interface for managing dataset columns. At the top, there are three input fields: 'Righe da campionare' (200), 'Considera come Null' (empty), and 'Forza troncamento spazio vuoto' (disabled). Below this is a table titled 'Colonne da caricare' (Load columns) with the following data:

Nome colonna	Tipo di colonna	Lunghezza massima	Formato di visualizzazione	Separatore di gruppo	Carattere decimale
CRASH_RECORD_ID	VARCHAR2	255			
CRASH_DATE_EST_I	VARCHAR2	1			
CRASH_DATE	DATE		MM"/"DD"/"YYYY" "HH"		
POSTED_SPEED_LIMIT	NUMBER				
TRAFFIC_CONTROL_DEVICE	VARCHAR2	50			

Figura 2.4: Modifica manuale del parsing automatico delle colonne del dataset

Una volta impostate tutte le caratteristiche, si completa la creazione della tabella. Quest’ultima può ora essere selezionata e visualizzata in dettaglio nel Browser Oggetti, la cui sezione DDL mostra il codice SQL di creazione generato in automatico dalla procedura (Figure 2.5 e 2.6).

```

1  CREATE TABLE "A01_TRAFFIC_ACCIDENTS"
2    (
3      "ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY
4          MINVALUE 1 MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1
5          START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,
6      "CRASH_RECORD_ID" VARCHAR2(255),
7      "CRASH_DATE_EST_I" VARCHAR2(1),
8      "CRASH_DATE" DATE,
9      "POSTED_SPEED_LIMIT" NUMBER,
10     "TRAFFIC_CONTROL_DEVICE" VARCHAR2(50),
11     "DEVICE_CONDITION" VARCHAR2(50),
12     "WEATHER_CONDITION" VARCHAR2(50),
13     "LIGHTING_CONDITION" VARCHAR2(50),
14     "FIRST_CRASH_TYPE" VARCHAR2(50),
15     "TRAFFICWAY_TYPE" VARCHAR2(255),
16     "LANE_CNT" NUMBER,
17     "ALIGNMENT" VARCHAR2(50),
18     "ROADWAY_SURFACE_COND" VARCHAR2(50),
19     "ROAD_DEFECT" VARCHAR2(50),
20     "REPORT_TYPE" VARCHAR2(50),
21     "CRASH_TYPE" VARCHAR2(255),
22     "INTERSECTION RELATED_I" VARCHAR2(1),
23     "NOT_RIGHT_OF WAY_I" VARCHAR2(1),
24     "HIT_AND_RUN_I" VARCHAR2(1),
25     "DAMAGE" VARCHAR2(50),
26     "DATE_POLICE_NOTIFIED" DATE,
27     "PRIM_CONTRIBUTORY_CAUSE" VARCHAR2(255),
28     "SEC_CONTRIBUTORY_CAUSE" VARCHAR2(255),
29     "STREET_NO" NUMBER,
30     "STREET_DIRECTION" VARCHAR2(1),
31     "STREET_NAME" VARCHAR2(100),
32     "BEAT_OF_OCCURRENCE" NUMBER,

```

Figura 2.5: Codice DDL di creazione della tabella A01_TRAFFIC_ACCIDENTS (prima parte)

```

32   "PHOTOS_TAKEN_I" VARCHAR2(1),
33   "STATEMENTS_TAKEN_I" VARCHAR2(1),
34   "DOORING_I" VARCHAR2(1),
35   "WORK_ZONE_I" VARCHAR2(1),
36   "WORK_ZONE_TYPE" VARCHAR2(50),
37   "WORKERS_PRESENT_I" VARCHAR2(1),
38   "NUM_UNITS" NUMBER,
39   "MOST_SEVERE_INJURY" VARCHAR2(50),
40   "INJURIES_TOTAL" NUMBER,
41   "INJURIES_FATAL" NUMBER,
42   "INJURIES_INCAPACITATING" NUMBER,
43   "INJURIES_NON_INCAPACITATING" NUMBER,
44   "INJURIES_REPORTED_NOT_EVIDENT" NUMBER,
45   "INJURIES_NO_INDICATION" NUMBER,
46   "INJURIES_UNKNOWN" NUMBER,
47   "CRASH_HOUR" NUMBER,
48   "CRASH_DAY_OF_WEEK" NUMBER,
49   "CRASH_MONTH" NUMBER,
50   "LATITUDE" NUMBER,
51   "LONGITUDE" NUMBER,
52   "LOCATION" VARCHAR2(255),
53   | PRIMARY KEY ("ID")
54   USING INDEX ENABLE
55   );

```

Figura 2.6: Codice DDL di creazione della tabella A01_TRAFFIC_ACCIDENTS (seconda parte)

2.2.2 Normalizzazione e tabelle di lookup

Oracle Database è un DBMS di tipo relazionale; ciò implica che i dati da utilizzare come base per realizzare gli applicativi debbano essere strutturati secondo un modello relazionale.

Per fare un breve rimando, il *modello relazionale* è un modello di struttura dei dati che si basa sul concetto matematico di relazione e che segue delle regole ben precise.

In ambito matematico, una relazione è definita come un sottoinsieme di un prodotto cartesiano, cioè un insieme di tuple ordinate rispetto ai relativi domini, rappresentanti gli insiemi di valori permessi (Figura 2.7).

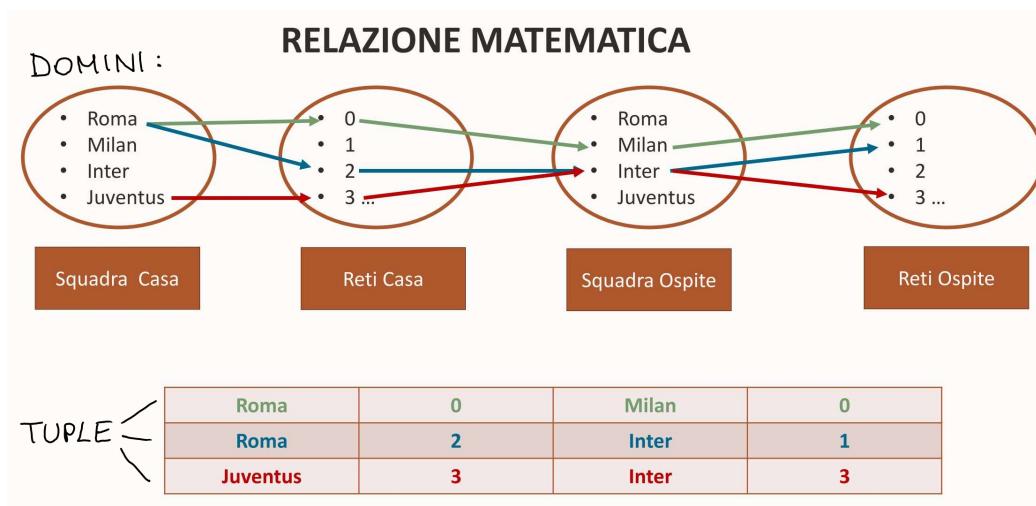


Figura 2.7: Schema esemplificativo di relazione matematica

Nel modello relazionale il concetto è analogo, ma la struttura non è posizionale: a ciascun dominio si associa un nome, detto attributo, che ne descrive il ruolo; con tupla, o record, si

indica la funzione che associa a ciascun attributo un valore del dominio (Figura 2.8).

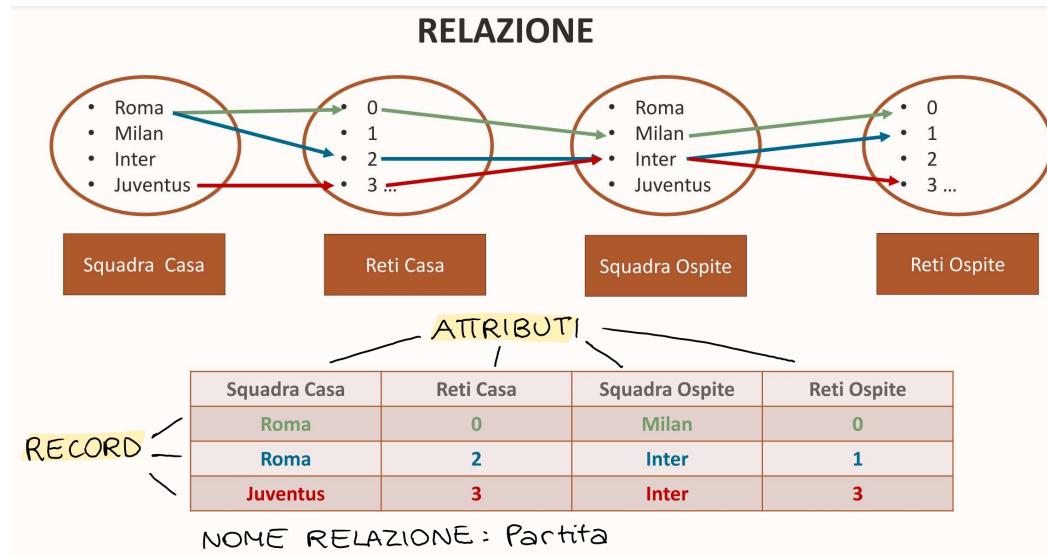


Figura 2.8: Relazione nel modello relazionale, con attributi (colonne) e record (righe)

Una tabella rappresenta una *relazione* se rispetta i seguenti requisiti:

- non ha righe nulle;
- non ha righe duplicate;
- i valori di ogni colonna sono fra loro omogenei.

I dati presenti in una relazione sono definiti *dati strutturati*, mentre i collegamenti tra più relazioni si dicono *associazioni*. Per comodità, da ora in avanti, con il termine "tabella" si indicherà sempre una relazione.

In virtù del fatto che non può contenere righe duplicate, una tabella ha sempre un numero minimo di attributi che individuano in maniera univoca ogni record. Questi attributi, o gruppi di attributi, sono chiamati *chiavi* e hanno le seguenti proprietà:

- *obbligatorietà*: non possono esistere record senza valore per una chiave;
- *unicità*: non possono esistere due record con lo stesso valore per una chiave;
- *non derivabilità*: i valori degli attributi chiave non devono poter essere calcolati a partire da altri attributi;
- *stabilità*: i valori assegnati alle chiavi devono rimanere immutabili nel tempo.

Si definisce *chiave primaria* di una relazione una tra tutte le sue chiavi candidate, scelta per essere usata come identificatore univoco dei record. Per semplicità, ove possibile, come chiave primaria si sceglie una chiave con un solo attributo; nel caso in cui non ci siano chiavi candidate con tale caratteristica, viene generalmente creata una chiave artificiale, detta *chiave primaria surrogata*, aggiungendo una nuova colonna alla tabella. Questo attributo, rinominato *id*, consiste solitamente in un contatore numerico che si autoincrementa ad ogni record aggiunto.

Due tabelle possono essere associate collegando uno o più attributi di una con uno o più attributi dell'altra. Le associazioni tipiche sono:

- uno a uno (1:1);

- uno a molti (1:N);
- molti a molti (N:N).

Ai fini pratici, l'associazione molti a molti è usata raramente. La più comune è, invece, quella uno a molti, nella quale la chiave primaria di una tabella è messa in relazione con uno o più attributi di un'altra. Gli attributi della tabella associata vengono chiamati *chiave esterna*. Questa associazione può essere soggetta ad un vincolo importante detto *vincolo di integrità referenziale*; esso garantisce che per ogni valore della chiave esterna esista un valore della chiave primaria nella tabella associata. In questo modo si evitano anomalie e incongruenze durante la memorizzazione dei dati.

La *normalizzazione* è il processo di organizzazione dei dati all'interno di un database volto a ridurre ridondanze e anomalie e a migliorare l'integrità delle informazioni. Per normalizzare un database è necessario seguire alcune regole chiamate *forme normali* (NF). Rispettando le prime tre regole, si dice che il database è in terza forma normale (3NF), considerata come il più alto requisito di progettazione necessario per la maggior parte delle applicazioni.

I passi per portare il database in 1NF sono i seguenti:

- eliminare i gruppi di attributi ripetuti in singole tabelle;
- creare una tabella separata per ciascun insieme di dati correlati;
- identificare univocamente ciascuna tabella con una chiave primaria.

Ciò significa che non bisogna utilizzare attributi multipli in una singola tabella per memorizzare dati simili.

Un esempio è dato da PRIM_CONTRIBUTORY_CAUSE e SEC_CONTRIBUTORY_CAUSE, i campi riferiti, rispettivamente, alle cause primaria e secondaria di un incidente stradale. Questi due attributi descrivono, infatti, lo stesso tipo di dato e contengono gli stessi valori: le possibili concuse di un incidente. Il problema insorge nel caso si voglia aggiungere, ad esempio, una terza concausa. Questa azione richiederebbe l'aggiunta di una terza colonna THIRD_CONTRIBUTORY_CAUSE e, di conseguenza, la modifica della struttura dell'intera tabella e di tutti i suoi record. A livello teorico sarebbe, quindi, opportuno creare altre due tabelle (Figura 2.9), ovvero:

1. una tabella contenente tutte le informazioni riguardanti le cause;
2. una tabella di collegamento rappresentante la relazione molti a molti tra la tabella originaria degli incidenti e quella delle cause.

The diagram illustrates the normalization process from 1NF to 3NF. It shows three tables:

- 1 tabella (1 table):** Contains columns Incidente_ID, Data, Causa1, Causa2, Causa3. Data contains dates (10/11/2024, 11/11/2024, 12/11/2024, 13/11/2024). Causa1, Causa2, and Causa3 contain categorical values (bicycle, weather, motorcycle, animal, cell phone use).
- 3 tabelle (3 tables):** Decomposes the original table into three separate tables. The first table has columns Incidente_ID and Data, with data matching the original table. The second table has columns Causa_ID and Causa, with data: Causa_ID 1 (animal), 2 (bicycle), 3 (cell phone use), 4 (weather), 5 (motorcycle). The third table has columns Incidente_ID and Causa_ID, linking the two decomposed tables.
- Tabella di collegamento (Linking table):** Shows the many-to-many relationship between Incidente_ID and Causa_ID. It has columns Incidente_ID, Causa_ID, and Ordine_Causa. The data shows multiple incidents (1, 2, 3, 4) each associated with multiple causes (1 through 5).

Figura 2.9: Esempio di normalizzazione in 1NF, con tabella dominio e tabella di collegamento N:N

Come scelta architetturale, delle volte, si può decidere di non normalizzare una struttura composta da pochi attributi, per evitare di sovra-ingegnerizzare e duplicare più volte le

dimensioni del database. Eccezioni come questa verranno comunque affrontate in fase di reporting con opportune viste di unione.

Basandosi la 2NF sulla presenza di chiavi composte da più attributi, che nel dataset in questione non compaiono, le sue condizioni coincidono con quelle della 3NF. Quest'ultima richiede di:

- creare tabelle separate per insiemi di valori validi per più record;
- correlare queste tabelle a una chiave esterna.

In generale, ogni volta che il contenuto di un gruppo di attributi può essere applicabile a più record della tabella, bisogna valutare la possibilità di inserire tali attributi in una tabella separata. Di seguito viene illustrato il processo di creazione di queste tabelle, chiamate *tabelle di lookup* o di ricerca.

Aprendo il riquadro di dettaglio della tabella principale A01_TRAFFIC_ACCIDENTS nell'area del Browser Oggetti, si va nella sezione 'Colonne' e si clicca, prima su 'Altro', poi su 'Crea tabella di ricerca'. Di conseguenza si aprirà una modale nella quale bisogna indicare la colonna dell'attributo che diverrà chiave esterna e il nome che si vuole dare alla tabella di lookup. Nel riquadro sottostante apparirà lo script SQL generato in automatico per compiere questa operazione. Nell'esempio in Figura 2.10, riguardante la tabella di lookup A01_LIGHTING_CONDITION, il codice esegue in ordine le seguenti azioni:

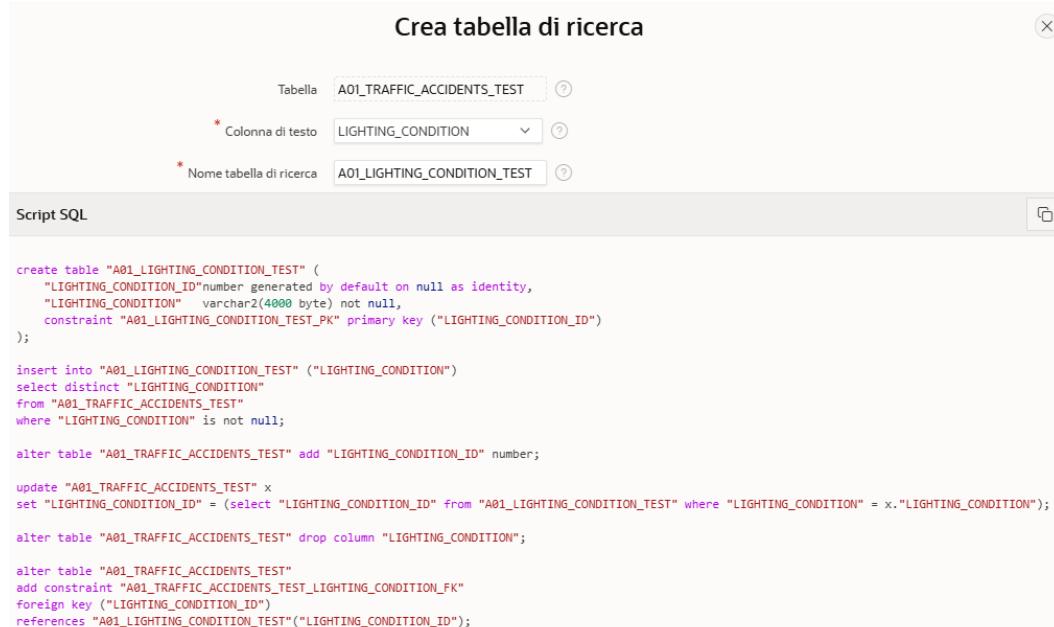


Figura 2.10: Esempio di creazione automatica di una tabella di lookup

- *creazione della tabella di lookup:* CREATE della tabella di lookup, con l'attributo autogenereato ID impostato come chiave primaria, e l'attributo testuale LIGHTING_CONDITION con il nome della condizione di luce;
- *popolamento della tabella di lookup:* INSERT nella tabella di lookup di tutti i valori distinti e non nulli dell'attributo LIGHTING_CONDITION della tabella principale;
- *aggiunta della nuova colonna per l'id nella tabella principale:* ALTER per inserire nella tabella principale la nuova colonna LIGHTING_CONDITION_ID, la quale farà da chiave esterna;

- *aggiornamento dei record*: UPDATE della tabella principale per popolare la nuova colonna e associare ogni record al relativo ID della tabella di lookup;
- *eliminazione della vecchia colonna testuale*: ALTER per eliminare la colonna originaria LIGHTING_CONDITION dalla tabella principale;
- *creazione della chiave esterna*: ALTER per inserire nella tabella principale il vincolo di chiave esterna da collegare alla chiave primaria della tabella di lookup.

Effettuando questo procedimento per ogni attributo di A01_TRAFFIC_ACCIDENTS che abbia valori che si ripetono nei record, si realizzano tutte le tabelle di lookup necessarie. Ognuna di queste genera, quindi, un vincolo di integrità referenziale con la tabella principale, la quale alla fine si ritrova ad avere la maggior parte delle colonne contenenti riferimenti numerici (Figura 2.11).

↓ Normalizzazione

Colonne	Dati	Indici	Vincoli	Privilegi	Statistiche	Trigger	Dipendenze	DDL	Query di esempio
+ Inserisci riga	Colonne...	Filtra...	Conta righe	Carica dati	Scarica	Aggiorna			
	ID	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE	DEVICE_CONDITION	WEATHER_CONDITION	LIGHTING_CONDITION	FIRST_CRASH_TYPE
31	545f783c7f29f20...	29/03/2021		10	NO CONTROLS	NO CONTROLS	CLEAR	DAYLIGHT	PARKED MOTOR VEHICLE
32	569af2dc0ea9c5...	18/09/2021		30	NO CONTROLS	NO CONTROLS	CLEAR	DAYLIGHT	SIDESWIPE SAME DIRECTION
33	57fcdd70be57a9...	12/08/2019		20	NO CONTROLS	NO CONTROLS	CLEAR	DAYLIGHT	REAR END
34	5bf5fb7b499fc13...	21/08/2022		15	NO CONTROLS	NO CONTROLS	CLEAR	DAYLIGHT	REAR END
35	5d2581779664d4...	20/07/2019		55	NO CONTROLS	NO CONTROLS	CLEAR	DAYLIGHT	REAR END
36	502985cd1c5282...	30/07/2023		30	NO CONTROLS	NO CONTROLS	CLEAR	DAYLIGHT	SIDESWIPE SAME DIRECTION
37	64063aa2537af63...	19/08/2023		30	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	SIDESWIPE SAME DIRECTION
38	5590930e0c8de5...	30/07/2023		20	NO CONTROLS	NO CONTROLS	UNKNOWN	UNKNOWN	PARKED MOTOR VEHICLE

Colonne	Dati	Indici	Vincoli	Privilegi	Statistiche	Trigger	Dipendenze	DDL	Query di esempio
+ Inserisci riga	Colonne...	Filtra...	Conta righe	Carica dati	Scarica	Aggiorna			
	ID	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE_ID	DEVICE_CONDITION_ID	WEATHER_CONDITION_ID	LIGHTING_CONDITION_ID	FIRST_CRASH_TYPE_ID
16	a5edde16c371ee2...	22/09/2023		30	18	7	10	3	1
17	3facd2ed99a27da...	27/09/2019		30	18	7	1	1	3
18	4502c89bb17d2a4...	19/02/2019		20	18	7	10	5	12
19	fd05285e9d273fe2...	29/07/2023		10	18	7	11	6	6
20	fda2491e353ac1904...	29/07/2023		30	8	1	10	5	7
21	c90531699ae6d24f...	22/09/2023		30	8	1	10	3	17
22	456dd99bc16f50...	24/05/2021		20	18	7	10	4	4

Figura 2.11: Contenuto della tabella principale dopo la creazione delle tabelle di lookup

2.2.3 Pulizia e vista di raggruppamento

Un aspetto importante da garantire all'interno del dataset è la qualità dei dati, che a volte risultano compromessi a causa di errori di input o di mancata manutenzione del database. Le diverse categorie di errore che possono presentarsi all'interno del database riguardano:

- dati mancanti;
- dati non corretti;
- valori non ragionevoli;
- formattazione inconsistente;
- errori di calcolo;
- dati che non rientrano nella loro codifica prevista.

Quello svolto sul dataset degli incidenti stradali non è stato un vero e proprio lavoro di pulizia profonda, quanto un insieme di interventi mirati di bonifica e standardizzazione dei dati, per migliorare la loro coerenza e usabilità. A tal proposito è stata creata, tramite script SQL, la vista A01_TRAFFIC_ACCIDENTS_VW che racchiude i seguenti interventi principali:

- *Gestione dei valori mancanti o sconosciuti:* ricodifica dei valori NULL, o non previsti, in categorie predefinite, come 'UNKNOWN' e 'OTHER'.
- *Standardizzazione semantica tramite raggruppamenti:* utilizzo della clausola SQL CASE WHEN per mappare singole categorie di valori in gruppi omogenei e sintetizzare le informazioni per i grafici (Figura 2.12). Esempi rilevanti a riguardo sono quelli relativi agli attributi first_crash_type, traffic_control_device e most_severe_injury.

```
case
when fct.first_crash_type in ('REAR END','REAR TO FRONT') then 'REAR-END'
when fct.first_crash_type in ('SIDESWIPE SAME DIRECTION','SIDESWIPE OPPOSITE DIRECTION') then 'SIDESWIPE'
when fct.first_crash_type in ('TURNING') then 'TURNING'
when fct.first_crash_type in ('ANGLE','REAR TO SIDE') then 'ANGLE'
when fct.first_crash_type in ('HEAD ON') then 'HEAD-ON'
when fct.first_crash_type in ('PARKED MOTOR VEHICLE') then 'PARKED VEHICLE'
when fct.first_crash_type in ('FIXED OBJECT','OTHER OBJECT') then 'OBJECT'
when fct.first_crash_type in ('PEDESTRIAN') then 'PEDESTRIAN'
when fct.first_crash_type in ('PEDALCYCLIST') then 'PEDALCYCLIST'
when fct.first_crash_type in ('REAR TO REAR','OTHER NONCOLLISION','OVERTURNED','ANIMAL','TRAIN') then 'OTHER'
end as first_crash_type_group,
```

Figura 2.12: Esempio di raggruppamento dei valori per l'attributo first_crash_type

- *Creazione di bucket per valori numerici:* i valori quantitativi di attributi numerici, come posted_speed_limit e lane_cnt, sono stati suddivisi in intervalli, chiamati bucket, per facilitare l'aggregazione e la rappresentazione grafica.
- *Gestione dei valori anomali:* l'attributo lane_cnt presentava valori non realistici, superiori anche a 1000. Dopo una ricerca sul numero massimo effettivo di corsie nelle strade di Chicago, tutti i valori di lane_cnt superiori a 12 sono stati inclusi nella categoria 'UNKNOWN/INVALID' (Figura 2.13).

```
ta.lane_cnt as num_road_lanes,
case
when ta.lane_cnt = 1 then '1'
when ta.lane_cnt = 2 then '2'
when ta.lane_cnt in (3,4) then '3-4'
when ta.lane_cnt >= 5 and ta.lane_cnt <= 12 then '5+'
else 'UNKNOWN/INVALID'
end as lane_bucket,
```

Figura 2.13: Creazione del bucket per l'attributo lane_cnt e gestione dei valori non validi

- *Formattazione testuale ed etichette per i grafici:* per garantire un corretto ordinamento e una rappresentazione coerente delle informazioni temporali nei grafici, sono stati effettuati dei TRIM, per rimuovere gli spazi superflui, e sono stati aggiunti dei prefissi numerici per gli attributi da mostrare nelle etichette (Figura 2.14). Alcuni esempi riguardano gli attributi crash_day_label, crash_month_label e crash_hour con l'arrotondamento orario.

La vista A01_TRAFFIC_ACCIDENTS_VW comprende, inoltre, le join con tutte le tabelle di lookup, per recuperare i valori descrittivi degli attributi (Figura 2.15). Come sarà illustrato nel prossimo capitolo, questa vista fungerà da sorgente dei dati per tutti i grafici presenti nell'applicativo.

```
-- TO CHAR per estrazione di certi dati da mostrare nei grafici
to_char(ta.crash_date, 'HH24:MI') as crash_time, -- 06:40 (già ordinabile)
trim(to_char(ta.crash_date, 'DAY', 'NLS_DATE_LANGUAGE=ENGLISH')) as crash_day, -- FRIDAY (trim per gli spazi alla fine)
to_char(ta.crash_date, 'DY', 'NLS_DATE_LANGUAGE=ENGLISH') as crash_day_short, -- FRI
trim(to_char(ta.crash_date, 'MONTH', 'NLS_DATE_LANGUAGE=ENGLISH')) as crash_month, -- SEPTEMBER
to_char(ta.crash_date, 'MON', 'NLS_DATE_LANGUAGE=ENGLISH') as crash_month_short, -- SEP
to_char(ta.crash_date, 'YYYY') as crash_year, -- 2023 (già ordinabile)

-- LABEL per i malfunzionamenti nell'ordinamento sui grafici a più serie
to_char(ta.crash_date, 'D') || ' - ' || trim(to_char(ta.crash_date, 'DAY', 'NLS_DATE_LANGUAGE=ENGLISH')) as crash_day_label, -- 5 - FRIDAY
to_char(ta.crash_date, 'MM') || ' - ' || trim(to_char(ta.crash_date, 'MONTH', 'NLS_DATE_LANGUAGE=ENGLISH')) as crash_month_label, -- 09 - SEPTEMBER
-- Label giorno (prefisso numerico + abbreviazione)
lpad(to_char(ta.crash_date, 'D'), 2, '0') || ' - ' || to_char(ta.crash_date, 'DY', 'NLS_DATE_LANGUAGE=ENGLISH') as crash_day_short_label, -- 05 - FRI
-- Label mese (prefisso numerico a 2 cifre + abbreviazione)
to_char(ta.crash_date, 'MM') || ' - ' || to_char(ta.crash_date, 'MON', 'NLS_DATE_LANGUAGE=ENGLISH') as crash_month_short_label, -- 09 - SEP

-- FORMATI NUMERICI per ordinamento
to_char(ta.crash_date, 'D') as crash_day_num, -- 5
to_char(ta.crash_date, 'MM') as crash_month_num, -- 09
-- Orario arrotondato
to_char(round(ta.crash_date, 'HH'), 'HH24') || ':00' as crash_hour, -- 07:00
```

Figura 2.14: Operazioni di TRIM e conversione in stringhe dei valori degli attributi temporali per il loro ordinamento e utilizzo nelle etichette dei grafici

```
create or replace view a01_traffic_accidents_vw as
> select ta.id, ...
  from a01_traffic_accidents ta

  left join a01_contributory_cause      pc on ta.prim_contributory_cause_id = pc.id
  left join a01_contributory_cause      sc on ta.sec_contributory_cause_id = sc.id
  left join a01_traffic_control_device tcd on ta.traffic_control_device_id = tcd.id
  left join a01_device_condition       dc on ta.device_condition_id = dc.id
  left join a01_weather_condition      wc on ta.weather_condition_id = wc.id
  left join a01_lighting_condition     lc on ta.lighting_condition_id = lc.id
  left join a01_roadway_surface_cond   rsc on ta.roadway_surface_cond_id = rsc.id
  left join a01_trafficway_type       tt on ta.trafficway_type_id = tt.id
  left join a01_alignment              a on ta.alignment_id = a.id
  left join a01_road_defect            rd on ta.road_defect_id = rd.id
  left join a01_first_crash_type      fct on ta.first_crash_type_id = fct.id
  left join a01_report_type           rt on ta.report_type_id = rt.id
  left join a01_crash_type             ct on ta.crash_type_id = ct.id
  left join a01_damage                d on ta.damage_id = d.id
  left join a01_street_direction      sd on ta.street_direction_id = sd.id
  left join a01_street_name            sn on ta.street_name_id = sn.id
  left join a01_work_zone_type        wzt on ta.work_zone_type_id = wzt.id
  left join a01_most_severe_injury    msi on ta.most_severe_injury_id = msi.id;
```

Figura 2.15: Elenco di tutte le operazioni di left join effettuate tra la tabella principale e le tabelle di lookup durante la creazione della vista

2.3 Applicativo APEX

Una volta completato il processo di elaborazione e di normalizzazione del dataset, si può procedere con la creazione dell'applicativo APEX. All'interno di quest'ultimo sono stati realizzati rispettivamente:

- un report interattivo con form per ogni tabella di lookup;
- una griglia interattiva con form per la tabella principale degli incidenti;
- cinque dashboard tematiche contenenti grafici interattivi;
- un menù di navigazione per spostarsi facilmente tra tutte le sezioni dell'applicativo e una home page che mostra l'elenco delle sezioni.

Di seguito saranno illustrati tutti i componenti dell'applicativo, al di fuori delle dashboard, che verranno trattate nel capitolo successivo.

2.3.1 Report interattivi delle tabelle di lookup

Per creare l'applicazione 'A01 Traffic Accidents' è stata utilizzata la procedura grafica guidata tramite wizard. Nella sezione App Builder dell'area di lavoro APEX si clicca su 'Crea' e si seleziona 'Usa Creazione guidata applicazione' per aprire la finestra delle impostazioni iniziali da fornire. Da qui si scelgono nome e tema dell'applicazione e si possono immediatamente creare delle pagine. Come prima cosa sono state realizzate delle pagine di tipo report interattivo, una per ogni tabella di lookup.

Nell'ambito degli applicativi APEX, il *report interattivo* è uno dei componenti principali e viene utilizzato per rappresentare in un certo formato tabellare i record restituiti da una query fatta sul database sorgente. Scegliendo come tipologia di pagina il report interattivo, si crea un'area di reporting che consente agli utenti di svolgere diverse azioni, tra cui:

- personalizzare e poi salvare il layout dei dati, scegliendo le colonne da visualizzare;
- utilizzare filtri ed ordinamenti avanzati;
- definire regole di formattazione condizionale;
- definire interruzioni, aggregazioni, grafici e colonne formula;
- scaricare in locale i dati in diversi formati.

Nel caso in questione, tutte le tabelle di lookup sono state rappresentate tramite report interattivo. La struttura del report è visualizzabile nel riquadro di sinistra del Page Designer (Figura 2.16) e si presenta come un elenco di "elementi colonna", dove la colonna dell'ID è nascosta di default.

All'interno dell'applicativo, il report appare come una tabella con a fianco ad ogni record un'icona di modifica. Cliccando sull'icona si apre una form modale a cassetto per modificare i valori degli attributi per il record selezionato. Cliccando, invece, sul pulsante 'Crea' in alto a destra, compare la form di inserimento di un nuovo record. Il report interattivo è, quindi, un formato estremamente comodo e semplice da utilizzare per la gestione e la visualizzazione dei dati all'interno di un sistema, come potrebbe essere quello del dipartimento di polizia.

The figure consists of three panels illustrating the APEX Page Designer and its runtime application.

- Panel a (Left):** Shows the Page Designer's left sidebar with navigation links like 'Precedente al rendering', 'Componenti', 'Breadcrumb Bar', 'Body', 'Report salvati', 'Azioni dinamiche', and 'A destra della barra di ricerca del report'. Below the sidebar is a 'CREATE' button.
- Panel b (Center):** Shows the 'Lighting Condition' report page. The title bar says 'Lighting Condition'. The page includes a search bar ('Q'), a toolbar with 'Vai' and 'Azioni' buttons, and a 'Crea' button in the top right. The main content area displays a table with columns 'Lighting Condition ↑' and 'Actions'. The table rows are:

DARKNESS	
DARKNESS, LIGHTED ROAD	
DAWN	
DAYLIGHT	
DUSK	
UNKNOWN	

 The bottom right corner of the page shows '1 - 6'.
- Panel c (Right):** Shows a modal dialog titled 'Lighting Condition' with an 'Obligatorio' label. It contains a single input field labeled 'Lighting Condition' with a red asterisk indicating it is required. At the bottom are 'Annulla' and 'Crea' buttons.

Figura 2.16: a) Riquadro di sinistra del Page Designer con la struttura del report. b) Visualizzazione del report all'interno dell'applicativo. c) Form di inserimento di un nuovo record

2.3.2 Griglia interattiva degli incidenti stradali

La griglia interattiva è un altro componente APEX analogo al report, ma meno utilizzato, anche se presenta alcuni vantaggi di visualizzazione in più. Sia il report che la griglia permettono di estrarre i dati in formato tabellare, di implementare filtri, di ordinare i campi e di salvare layout personalizzati. La sostanziale differenza tra i due consiste nel fatto che in una griglia interattiva si possono modificare i dati direttamente nella tabella, senza la necessità di una modale con la form. Dal punto di vista della visualizzazione, poi, la griglia permette di ordinare a piacimento le colonne tramite drag and drop.

Per i vantaggi appena spiegati, si è scelto di rappresentare la vista, contenente i dati puliti e rielaborati, tramite griglia interattiva (Figura 2.17) munita di form per la creazione e la modifica dei record degli incidenti stradali (Figura 1.15). Nella Figura 2.18 si possono osservare degli esempi di azioni di filtraggio, aggregazione e conteggio dei record e degli attributi del dataset.

Figura 2.17: Griglia interattiva della vista, con l'elenco di tutte le sue azioni

Figura 2.18: Filtri, aggregazioni e conteggi nella griglia interattiva

2.3.3 Home Page e menù di navigazione

Il menù di navigazione è un componente dell'applicativo APEX che possiede un ruolo fondamentale, poiché aiuta l'utente a capire la struttura dell'app e gli consente di navigare efficacemente tra i vari contenuti. Esso viene aggiunto in automatico dal sistema durante la creazione di un applicativo e può essere aggiornato manualmente, o in maniera guidata, quando si aggiungono nuove pagine.

Il menù di navigazione è un esempio di lista. In Oracle APEX una *lista* è un oggetto che serve per definire un elenco di collegamenti verso altre pagine dell'applicativo o verso URL esterni. Ciascun elemento della lista è chiamato *voce di lista* e possiede diverse proprietà come, ad esempio, un'etichetta, un'icona e un URL di destinazione.

Il menù di navigazione si trova nell'App Builder, sotto la sezione 'Componenti condivisi' dell'applicazione. Da questa sezione, cliccando sulla voce 'Menu di navigazione', si apre una schermata che mostra l'elenco di tutte le sue voci di lista. Tra le proprietà delle voci di lista del menù si evidenziano:

- *la sequenza*: un valore numerico per stabilire la priorità della voce nell'ordinamento per la visualizzazione;
- *l'icona*: un'icona del tema scelto da poter eventualmente aggiungere a fianco al nome;
- *il nome*: da mostrare nella barra di navigazione;
- *la destinazione*: riferita alla pagina di destinazione che viene aperta cliccando sulla voce del menù.

Durante la creazione di una nuova pagina tramite la procedura grafica è sempre possibile definire tutte le opzioni legate alla navigazione. Tra queste, c'è la possibilità di creare una nuova voce di lista per includere nel menù la pagina che si sta creando, oltre che a selezionare un'eventuale voce padre che la comprende.

Il menù di navigazione è sempre accessibile all'interno dell'applicativo cliccando sull'*icona hamburger* in alto a sinistra (Figura 2.19).

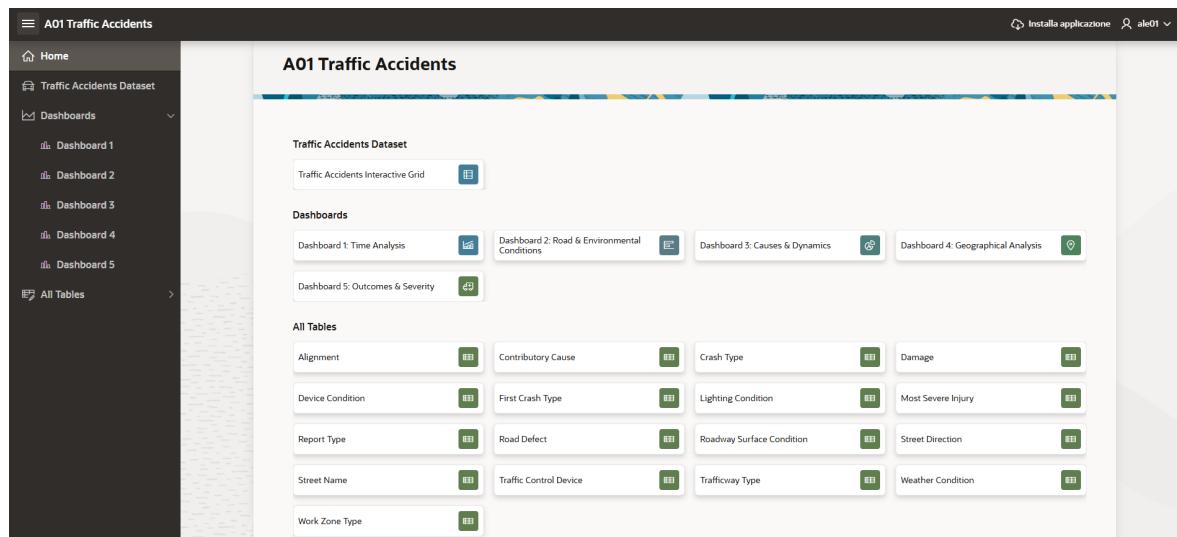


Figura 2.19: Home Page dell'applicativo con menù di navigazione

In questo caso sono presenti quattro voci di menù principali, ovvero:

1. *Home*: contiene il link alla Home Page;

2. *Traffic Accidents Dataset*: contiene il link alla pagina con la griglia interattiva degli incidenti stradali;
3. *Dashboards*: contiene altre cinque voci di menù ‘figlie’, una per ciascuna pagina, con una dashboard;
4. *All Tables*: contiene altre voci figlie, una per ciascuna pagina, con una tabella di lookup.

La Home Page è la prima schermata dopo l’accesso; essa contiene tre aree di tipo ‘Lista’ che replicano le voci del menù per i collegamenti a tutte le pagine.

CAPITOLO 3

Progettazione e implementazione delle dashboard di base

In questo capitolo verrà presentata la teoria alla base della realizzazione dei grafici per la visualizzazione interattiva dei dati del dataset. Saranno illustrati gli obiettivi principali dell'analisi per il supporto alle decisioni e i relativi strumenti. In seguito, si discuteranno le scelte effettuate sulla tipologia dei grafici e sulla loro organizzazione in cinque dashboard tematiche, al fine di rendere la consultazione dei dati più leggibile e coerente.

3.1 Obiettivi di analisi e criteri di design

In questa sezione verranno introdotti gli obiettivi di analisi e i principi di design adottati per progettare le dashboard. Definendo in anticipo queste regole si può garantire una coerenza tra le domande di analisi, le scelte di visualizzazione dei dati e la struttura delle pagine dell'applicativo, in modo che ogni grafico abbia un ruolo ben preciso e fornisca risultati chiari e confrontabili.

3.1.1 Data analytics e DDDM

La *data analytics* è una vasta disciplina che comprende l'analisi dei dati e che include la gestione del ciclo di vita completo delle informazioni: raccolta, pulizia, organizzazione, memorizzazione, analisi e governo dei dati. Con quest'ultimo termine si fa riferimento all'insieme di regole e processi per garantire la qualità, la sicurezza e la tracciabilità del dato.

Il processo decisionale basato sui dati, riferito in inglese come *Data Driven Decision Making* (DDDM), è l'approccio che utilizza i risultati forniti dall'analitica dei dati per guidare al meglio le procedure decisionali all'interno di realtà aziendali e organizzative. La data analytics, quindi, potenzia il DDDM con una base scientifica, in modo tale che le decisioni siano basate su dati di fatto e non semplicemente sull'esperienza passata o sul solo intuito.

Ci sono quattro categorie generali di data analytics che si distinguono sulla base dei risultati prodotti:

- *Analitica descrittiva* (Cosa è successo?): riassume e descrive i dati passati per comprendere le caratteristiche di base. Questa analisi viene spesso effettuata tramite dei sistemi di reporting o dei set di dashboard.

- *Analitica diagnostica* (Perché è successo?): individua i modelli e le tendenze principali osservati nell’analisi descrittiva. I risultati di questa analisi vengono solitamente esplorati tramite strumenti di visualizzazione interattiva per identificare pattern.
- *Analitica predittiva* (Cosa succederà?): utilizza i dati storici, la modellazione statistica e l’apprendimento automatico per fare previsione su tendenze ed eventi futuri. Questa analisi viene utilizzata nell’identificazione di rischi e opportunità.
- *Analitica prescrittiva* (Cosa fare?): va oltre la previsione e fornisce raccomandazioni per l’ottimizzazione delle azioni future, basandosi sui risultati derivati da tutte le analisi precedenti.

La qualità dell’intero processo appena descritto dipende dalla coerenza tra la struttura dei dati, gli obiettivi e il metodo utilizzati. Per questo, oltre ad un corretto processo di raccolta ed elaborazione delle informazioni (ETL), è importante definire uno scopo ben preciso per l’analisi.

Nel caso di questo progetto, l’analisi proposta esplora schemi ricorrenti nel tempo e nello spazio e mette in relazione cause e condizioni degli incidenti con i loro esiti, offrendo indicazioni utili per la sicurezza stradale, la pianificazione urbana e la valutazione delle politiche pubbliche. In particolare, sono state impiegate analitiche descrittive e diagnostiche per trovare pattern e relazioni utili, mentre una componente predittiva è stata implementata come estensione per fare delle previsioni e verrà trattata nel capitolo successivo.

3.1.2 Domande di analisi

L’analisi degli incidenti stradali, in particolare, si colloca all’incrocio fra sicurezza pubblica, pianificazione urbana e governo del traffico. L’obiettivo, in questo caso, non è solo quello di descrivere quante collisioni avvengono, ma di comprendere quando e dove si concentrano, quali condizioni le accompagnano e con quale severità si manifestano. Le domande tipiche da prendere in considerazione durante la definizione del processo di analisi e durante la successiva progettazione delle dashboard sono:

- *Quando?* Per descrivere picchi giornalieri e settimanali e tendenze per anno e mese.
- *Dove?* Per rilevare aree e tratti stradali con una maggiore incidenza.
- *Per quali cause?* Per studiare le dinamiche e i fattori che hanno contribuito.
- *Con quali condizioni?* Per individuare le principali condizioni in cui avvengono gli incidenti.
- *Con quali esiti?* Per studiare la severità degli incidenti e il suo andamento temporale.

Come primo approccio di generazione di risultati, sono state realizzate cinque dashboard tematiche: la prima implementa un’analisi descrittiva, mentre le altre quattro adottano un’impostazione più diagnostica.

3.1.3 Linee guida per la visualizzazione dei dati

La *data visualization* è la rappresentazione grafica delle informazioni attraverso elementi visivi come diagrammi, grafici e mappe. Questi strumenti aiutano a spiegare i contenuti di un’analisi, organizzando i dati in maniera più comprensibile e mettendo in evidenza tendenze, ricorrenze e valori anomali. Una visualizzazione efficace del dataset consente di

esporlo in maniera coerente e facilmente consultabile, eliminando informazioni superflue e portando in primo piano quelle utili.

Un’importante premessa da fare riguardo alla data visualization è che i grafici non sono tutti uguali, ma ognuno di essi è più adatto a mostrare particolari tipi di relazioni rispetto ad altri. Prima di esaminare i diversi tipi di grafico a disposizione, è necessario studiare bene le domande di analisi accennate in precedenza, in modo da comprendere meglio i dati che si hanno a disposizione e, quindi, scegliere il miglior tipo di grafico per rappresentarli. Con particolare riferimento alla visualizzazione dei dati, le domande utili da considerare sono le seguenti:

- *Quale storia si sta cercando di raccontare?* Bisogna capire l’obiettivo del lavoro che si vuole svolgere con i dati e comprendere bene i concetti che essi descrivono.
- *A chi verranno presentati i risultati?* È importante conoscere bene il pubblico che leggerà i risultati del lavoro, il quale può essere composto da persone che conoscono bene l’argomento e che devono esplorarlo in dettaglio, oppure da persone meno informate, che necessitano di una visione più generale.
- *In che modo i diversi elementi dei dati sono correlati tra loro?* È utile comprendere come i dati si relazionano fra loro, cioè se variano nel tempo, e sono quindi assimilabili ad una serie temporale, o se invece sono più una distribuzione.
- *Che cosa si vuole mostrare?* È efficace scegliere in anticipo i principali elementi da mostrare nei grafici, come titoli, etichette, legende ed eventuali descrizioni.

In linea generale è possibile suddividere i grafici in macrocategorie, in base ai dati che presentano e alla modalità con cui li espongono (Figura 3.1). Tali macrocategorie sono:

- *Distribuzione:* sono i grafici delle serie temporali; questi sono utili per mostrare la stagionalità, i cicli e le tendenze di dati variabili nel tempo. Alcuni esempi utilizzati in questo progetto sono i grafici a barre verticali e i grafici a linee.
- *Comparazione:* sono i grafici per il confronto di dati appartenenti a categorie differenti, al fine di identificare differenze, somiglianze e tendenze. Per queste analisi si utilizzano solitamente grafici a barre, orizzontali o verticali, grafici a linee e grafici a radar.
- *Composizione:* sono i grafici utilizzati per analizzare come le varie parti compongono il tutto; alcuni esempi sono il diagramma a torta, il diagramma a ciambella e il grafico a barre impilate e impilate 100%.
- *Relazione:* sono i grafici utili per definire le relazioni tra i dati di categorie diverse; solitamente si utilizzano diagrammi di dispersione o grafici combinati a barre e linee.

La progettazione dei grafici per la visualizzazione dei dati è stata fatta seguendo il principio di creazione di un grafico dedicato per ogni domanda di analisi. Per garantire coerenza visiva e accessibilità delle informazioni, sono state adottate delle etichette sintetiche e ordinabili, delle legende uniformi e una paletta cromatica diversa per ogni dashboard. Inoltre, le interazioni con i grafici sono fluide, grazie ai filtri e alle modali di dettaglio, mentre le prestazioni restano buone; questo perché tutta la logica, compresa di raggruppamenti e aggregazioni, viene eseguita lato SQL nella vista analitica A01_TRAFFIC_ACCIDENTS_VW e nelle query sorgenti che la richiamano.

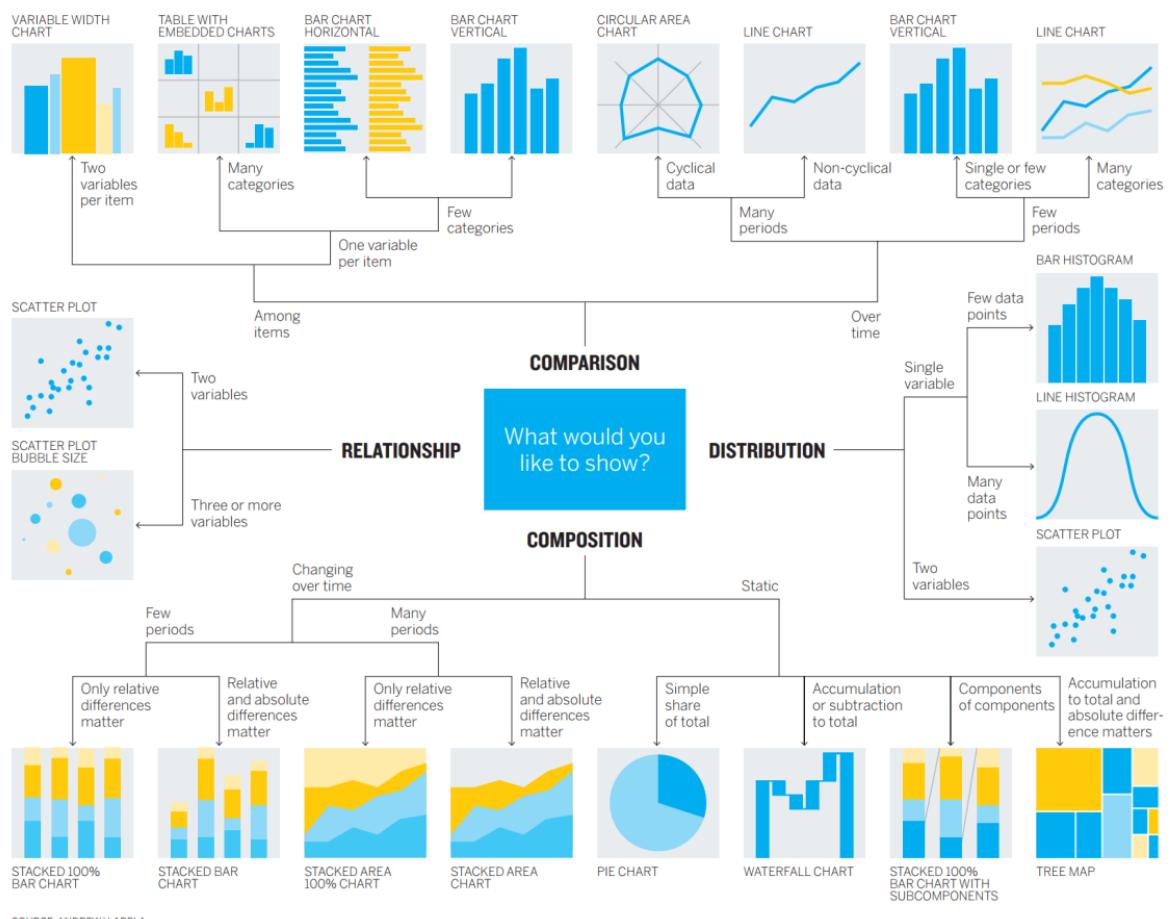


Figura 3.1: Schema di tutte le tipologie di grafici suddivisi per macrocategorie

3.2 Componenti comuni

Prima di entrare nel dettaglio delle singole dashboard, verranno descritti tutti i loro principali componenti, dalla sorgente dati alla struttura delle pagine, adottata per garantire una consistenza visiva e una facile consultazione.

3.2.1 Sorgente dei dati e mappatura delle colonne

In Oracle APEX, ogni grafico prende i dati necessari da una query SQL e mappa le colonne restituite sui seguenti campi del componente:

- *Nome serie*: identifica le serie, cioè categorie distinte di raggruppamento dei dati, che si vogliono visualizzare nel grafico. Tutti i record con lo stesso valore per una di queste colonne appartengono alla stessa serie. Nei grafici a singola serie questo campo può essere omesso o impostato come valore costante. Per dare un esempio, in Figura 3.2 appare l'editor delle proprietà per il grafico multi-serie Crashes by Month and Year, dove ogni anno è una serie distinta (2016, ..., 2023); l'effetto, osservabile in Figura 3.3, è la visualizzazione, per ogni mese, di più segmenti, uno per anno, impilati nella stessa colonna.
- *Etichetta*: è la categoria di raggruppamento scelta per essere rappresentata sull'asse x.

- *Valore*: è la misura numerica visualizzata, come può essere un conteggio, una somma o una media. L'eventuale aggregazione dei valori può essere gestita nella query sorgente o lasciata al motore del grafico, come nel caso del calcolo automatico della percentuale nei diagrammi a torta o a ciambella.

La query sorgente può opzionalmente fornire colonne aggiuntive per impostare classi CSS per colori e icone o, nel caso dei filtri dinamici, elementi di pagina da sottomettere per l'aggiornamento AJAX del grafico.

The screenshot shows the 'Origine' (Source) section of the Page Designer. It includes:

- Posizione**: Database locale
- Tipo**: Query SQL
- Query SQL** (Content):

```
select crash_month_num,
       crash_year,
       count(*) as crash_count
  from a01_traffic_accidents_vw
 where crash_year between 2016 and 2023
group by crash_month_num,
         crash_year
order by crash_month_num,
         crash_year;
```
- Elementi pagina da sottomettere**: An empty list.
- Suggerimento ottimizzatore**: An empty list.
- Order By**: Tipo: Nessuno
- Mapping colonne** (Content):

Nome serie	CRASH_YEAR
Etichetta	CRASH_MONTH_NUM
Valore	CRASH_COUNT

Figura 3.2: Sezioni del Page Designer con la query sorgente e il mapping delle colonne per il grafico *Crashes by Month and Year* della Dashboard 1

3.2.2 Pattern di pagina

Tutte le dashboard sono state progettate cercando di seguire dei principi per trovare il giusto equilibrio tra estetica e funzionalità. Attraverso i grafici, i dati devono essere mostrati come informazioni chiare e in una forma che catturi l'attenzione, risponda ai bisogni degli utenti e renda le decisioni più rapide e consapevoli. Come prima cosa, è importante definire lo scopo preciso per ciascuna dashboard, in modo da poterla classificare in una delle seguenti categorie:

- *strategica*: offre una visione d'insieme con KPI chiave per delineare strategie;
- *operativa*: monitora processi e performance quotidiane;
- *analitica*: analizza dati in dettaglio per individuare tendenze o anomalie.

È inoltre buona norma evitare di sovraccaricare la dashboard con troppe informazioni o elementi decorativi non necessari; bisogna, invece, cercare di mostrare solo i dati essenziali, mantenendo un design semplice ma efficace. A tal proposito, è stato scelto per le dashboard in questione il seguente pattern strutturale:

1. *Intestazione con KPI*: gli indici KPI, raccolti in un report classico con layout a ‘Cards’, sono disposti, in gruppi da quattro, all’inizio di ciascuna dashboard.
2. *Sezione dei grafici*: corrisponde al corpo centrale della dashboard e raccoglie dai cinque ai sei grafici; questi sono disposti in maniera ordinata lungo la struttura a righe e colonne della pagina.
3. *Modale di dettaglio*: il click su una qualsiasi porzione di grafico apre una modale contenente la griglia interattiva degli incidenti, pre-filtrata in base ai parametri selezionati.
4. *Palette di colori dedicata*: ogni dashboard è associata a una distinta palette di colori dell’Oracle Universal Theme, in modo da facilitarne visivamente il riconoscimento.

3.3 Dashboard tematiche interattive

In questa sezione verranno illustrate nel dettaglio le cinque dashboard tematiche progettate per condurre analisi descrittive e diagnostiche. I grafici presentano i dati rielaborati opportunamente per mostrare conteggi, percentuali, tendenze, distribuzioni e incroci tra fattori ed esiti degli incidenti.

È importante puntualizzare che le query di origine di alcuni grafici applicano alla vista principale A01_TRAFFIC_ACCIDENTS_VW delle aggregazioni aggiuntive e delle selezioni mirate, al fine di mantenere le visualizzazioni chiare e non sovraccaricate. Certe analisi, infatti, utilizzano categorie con molti valori, come, ad esempio, quelle riguardanti le cause e le tipologie di incidente; altre, invece, richiedono classifiche e selezione dei primi Top-n valori, come la classifica delle strade più pericolose. Ciò implica l’utilizzo di ulteriori raggruppamenti di valori e di istruzioni come RANK/DENSE RANK e FETCH FIRST n ROWS nelle query di origine. La soluzione architettonale migliore sarebbe quella di realizzare una vista dedicata per ciascun grafico; per semplicità, in diversi casi si è optato per una CTE (Common Table Expression) con la clausola WITH, ossia una vista temporanea nella query sorgente per selezionare e formattare esattamente i dati necessari a quel grafico. Nelle prossime sezioni, ciascuna dedicata ad una dashboard, verranno analizzati i grafici e le loro query sorgenti nel dettaglio.

3.3.1 Dashboard 1: Analisi temporale

Lo scopo scelto per la prima dashboard è quello di evidenziare l’andamento complessivo degli incidenti, la stagionalità, i picchi settimanali, gli orari e le variazioni anno su anno. I campi della vista utilizzati per questa analisi sono stati quelli riguardanti anno, mese, giorno della settimana e orario dell’incidente.

La struttura rispetta il pattern definito in precedenza, il quale presenta, come primo elemento, nell’intestazione, i KPI (Figura 3.3).

Gli indici KPI, acronimo di Key Performance Indicator, sono degli indicatori numerici che consentono di valutare l’andamento e i progressi di un processo aziendale. La scelta di KPI efficienti si basa su alcune caratteristiche fondamentali che questi devono avere. Essi, in primo luogo, devono essere rilevanti e significativi per l’analisi scelta; in secondo luogo, devono essere quantificabili, ovvero misurabili in termini matematici; infine, devono essere continuativi, ovvero misurati con opportuna periodicità.

Nell’ambito della sicurezza stradale e, più in generale, della gestione dei rischi, i KPI fungono sia da sistema di avviso preventivo sia da piano di crescita per l’organizzazione, identificando i potenziali rischi e le opportunità di miglioramento.



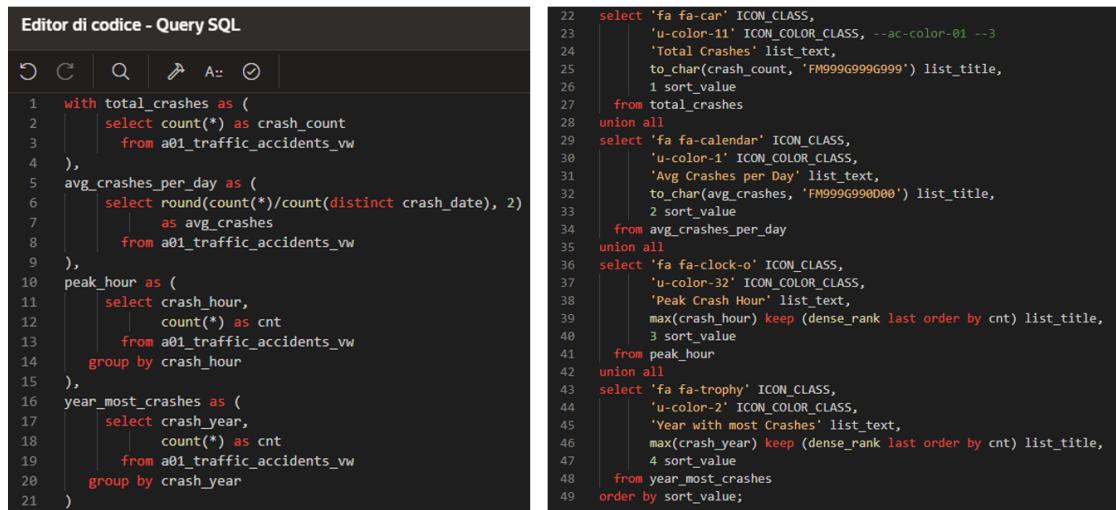
Figura 3.3: Pagina dell’applicativo APEX contenente la Dashboard 1 sull’analisi temporale

Per questa prima dashboard, focalizzata sull’analisi dell’andamento temporale degli incidenti, i KPI scelti sono i seguenti:

1. *Total Crashes*: numero complessivo di incidenti registrati nel dataset; utile per comprendere la dimensione del fenomeno e per fare da riferimento nelle analisi successive.
2. *Avg Crashes per Day*: media giornaliera degli incidenti; utile soprattutto per fare confronti omogenei tra periodi aventi una diversa copertura dei dati.
3. *Peak Crash Hour*: ora con la massima frequenza di incidenti; utile per evidenziare fasce orarie critiche, in modo da fare prevenzione e interventi mirati.
4. *Year with most Crashes*: anno in cui si è registrato il maggior numero di incidenti; utile come riferimento immediato per le analisi year-over-year (YoY) e per studiare l’insorgenza di eventuali nuove cause e condizioni per gli incidenti.

Tutti i KPI delle dashboard vengono calcolati direttamente nella query di origine di un report classico di tipo ‘Cards Container’, utilizzando la vista A01_TRAFFIC_ACCIDENTS_VW. La struttura del codice SQL in Figura 3.4 è comune a tutte le dashboard.

Il codice presenta una CTE per ogni KPI, ossia una funzione per definire un risultato temporaneo da utilizzare per il calcolo dell’indice. Dopo la definizione delle CTE, nella stessa query, vi sono le SELECT.



```

Editor di codice - Query SQL

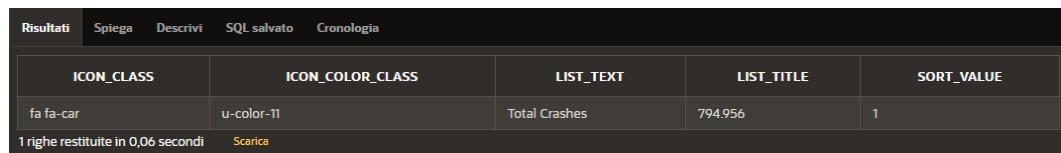
1 with total_crashes as (
2     select count(*) as crash_count
3         from a01_traffic_accidents_vw
4 ),
5 avg_crashes_per_day as (
6     select round(count(*)/count(distinct crash_date), 2)
7         as avg_crashes
8         from a01_traffic_accidents_vw
9 ),
10 peak_hour as (
11     select crash_hour,
12         count(*) as cnt
13         from a01_traffic_accidents_vw
14     group by crash_hour
15 ),
16 year_most_crashes as (
17     select crash_year,
18         count(*) as cnt
19         from a01_traffic_accidents_vw
20     group by crash_year
21 )
22 select 'fa fa-car' ICON_CLASS,
23     'u-color-11' ICON_COLOR_CLASS, --ac-color-01 --3
24     'Total Crashes' list_text,
25     to_char(crash_count, 'FM999G999G999') list_title,
26     1 sort_value
27     from total_crashes
28 union all
29 select 'fa fa-calendar' ICON_CLASS,
30     'u-color-1' ICON_COLOR_CLASS,
31     'Avg Crashes per Day' list_text,
32     to_char(avg_crashes, 'FM999G999D00') list_title,
33     2 sort_value
34     from avg_crashes_per_day
35 union all
36 select 'fa fa-clock-o' ICON_CLASS,
37     'u-color-32' ICON_COLOR_CLASS,
38     'Peak Crash Hour' list_text,
39     max(crash_hour) keep (dense_rank last order by cnt) list_title,
40     3 sort_value
41     from peak_hour
42 union all
43 select 'fa fa-trophy' ICON_CLASS,
44     'u-color-2' ICON_COLOR_CLASS,
45     'Year with most Crashes' list_text,
46     max(crash_year) keep (dense_rank last order by cnt) list_title,
47     4 sort_value
48     from year_most_crashes
49 order by sort_value;

```

Figura 3.4: Query sorgente per le card del report classico dei KPI della Dashboard 1

Ciascuna select restituisce una singola riga (Figura 3.5) composta dai seguenti campi:

- *LIST_TITLE*: titolo della card, contenente il valore del KPI;
- *LIST_TEXT*: sottotitolo della card, contenente la descrizione di riferimento del KPI;
- *ICON_CLASS*: classe CSS dell'icona da mostrare nella card;
- *ICON_COLOR_CLASS*: classe CSS di colore applicata alla card;
- *SORT_VALUE*: valore numerico intero utilizzato per l'ordinamento visivo delle card.



Risultati	Spiega	Descrivi	SQL salvato	Cronologia
ICON_CLASS	ICON_COLOR_CLASS	LIST_TEXT	LIST_TITLE	SORT_VALUE
fa fa-car	u-color-11	Total Crashes	794.956	1

1 righe restituite in 0,06 secondi [Scarica](#)

Figura 3.5: Risultato di una singola query per un KPI

Una volta impostata la query di origine del componente di tipo ‘Report Classico’, quest’ultimo mappa i valori dei cinque campi del risultato per effettuare il rendering delle card. La dashboard prosegue mostrando cinque grafici, di distribuzione e comparazione, dotati di legende interattive e pagine modali con la griglia pre-filtrata degli incidenti (Figura 3.6).



Figura 3.6: Apertura della pagina modale con la griglia degli incidenti pre-filtrata

Partendo dalla sezione in alto a sinistra e proseguendo verso destra, i grafici sono i seguenti:

1. *Cumulative Crashes by Year*: grafico combinato che mostra il conteggio annuale degli incidenti tramite barre e il valore cumulato tramite una linea. Si tratta di una distribuzione, per la tendenza annuale degli incidenti, e di una comparazione, per il confronto tra i singoli anni. Il codice sorgente è mostrato nella Figura 3.7 sottostante; il valore del grafico a barre (`crash_count`) è il conteggio degli incidenti raggruppati per anno, mentre il valore del grafico a linea (`crash_cumulative`) è la somma progressiva anno su anno. La cumulata è calcolata con una funzione analitica (`SUM(count(*)) OVER (ORDER BY crash_year)`), che consente di ottenere il totale progressivo senza ulteriori sottoquery.

```

1  select crash_year,
2      count(*) as crash_count,
3      sum(count(*)) over (order by crash_year) as crash_cumulative
4  from a01_traffic_accidents_vw
5  where crash_year between 2016 and 2023
6  group by crash_year
7  order by crash_year;

```

Figura 3.7: Query sorgente del grafico *Cumulative Crashes by Year*

Il grafico marca una crescita rapida degli incidenti tra il 2016 e il 2019, un improvviso calo nel 2020, e, infine, una ripresa nel 2021.

2. *Crashes by Month and Year*: grafico a barre verticali impilate a serie multiple, una per ogni anno, che confronta i volumi mensili degli incidenti. Come nel grafico precedente, anche qui si tratta di una distribuzione, riferita alla stagionalità mensile, e di una comparazione, per osservare gli scostamenti annuali nello stesso mese. Si notano picchi in settembre e ottobre e livelli più bassi nei mesi invernali e all'inizio di quelli primaverili.
3. *Crashes by Day of Week*: grafico a radar che confronta il numero di incidenti per giorno della settimana. È una comparazione che mostra valori variabili lungo il corso dell'intera settimana, con venerdì come giorno di picco e domenica come quello con valori più contenuti.
4. *Hourly Crash Distribution by Day of Week*: grafico a linee a serie multiple, una per ogni giorno della settimana. Si tratta di una distribuzione oraria con comparazione tra giorni; si rilevano doppi picchi nelle fasce orarie 08:00-09:00 e 16:00-18:00.
5. *Year Over Year Crash Count Change*: grafico combinato con barre verticali per la variazione assoluta (Delta Crash Count) del conteggio di incidenti anno su anno, e con linee con area per la variazione percentuale (Delta Crash Count %). È una comparazione di differenze di conteggio nel tempo, che mostra, per ogni anno, la variazione rispetto al precedente. Nella query sorgente, mostrata in Figura 3.8, la CTE `crash_counts` usa la funzione analitica `LAG(count(*)) OVER (ORDER BY crash_year)` per ottenere il totale di incidenti dell'anno prima; di conseguenza, nella CTE `crash_deltas` si derivano la variazione assoluta `delta_crash_count` e la variazione percentuale `delta_perc_crash_count`, gestendo anche il caso `prev_crash_count = 0` con COALESCE. L'intervallo è filtrato tra il 2016 e il 2023 poiché gli altri anni avevano una copertura dati troppo ridotta per essere confrontati. Il grafico mostra variazioni positive fino al 2018, l'anno di picco, poi negative nel 2020, fino a rilevare un rimbalzo nel 2021.

```

1  with crash_counts as (
2      select crash_year,
3          count(*) as crash_count,
4          coalesce(lag(count(*)) over ( order by crash_year ), 0) as prev_crash_count
5      from a01_traffic_accidents_vw
6      where crash_year between 2016 and 2023
7  group by crash_year
8  ),
9  crash_deltas as (
10     select crash_year,
11         crash_count,
12         prev_crash_count,
13         crash_count - prev_crash_count as delta_crash_count,
14         case
15             when prev_crash_count = 0
16             then 0
17             else round(100 * (crash_count - prev_crash_count) / prev_crash_count, 2)
18         end as delta_perc_crash_count
19     from crash_counts
20  )
21  select crash_year,
22         crash_count,
23         prev_crash_count,
24         delta_crash_count,
25         delta_perc_crash_count,
26         case
27             when delta_crash_count > 0 then '#bd4332' -- '#CA4D3C' -- rosso
28             else '#4190AC' -- '#577346' -- verde
29         end as delta_series_color,
30         case
31     from crash_deltas
32  order by crash_year

```

Figura 3.8: Query sorgente del grafico Year Over Year Crash Count Change

Tutte le osservazioni emerse da questa analisi costituiscono un supporto concreto per la pianificazione dei turni e per l’attuazione di interventi mirati durante le fasce orarie e i periodi più critici.

3.3.2 Dashboard 2: Condizioni ambientali e stradali

La seconda dashboard analizza come le condizioni ambientali e stradali influiscono sul numero e sulla severità degli incidenti (Figura 3.9). I campi della vista impiegati riguardano meteo, illuminazione, superficie della carreggiata, tipologie e stato dei dispositivi di controllo del traffico, numero di corsie e difetti stradali.

I KPI calcolati sono i seguenti:

1. *Crashes in Rain*: numero di incidenti avvenuti sotto la pioggia; misura l’impatto della condizione meteo più frequente.
2. *Night Crashes*: numero di incidenti notturni; utile per valutare rischi legati ad una ridotta visibilità e a scarsa illuminazione.
3. *Crashes due to Defective Road Surface*: incidenti avvenuti in strade con difetti della superficie; evidenzia possibili criticità correlate alla manutenzione delle infrastrutture stradali.
4. *Crashes with Malfunctioning Devices*: incidenti con dispositivi di controllo malfunzionanti; indica i volumi degli eventuali interventi richiesti sugli impianti di segnaletica.

Per questa dashboard sono stati realizzati cinque grafici, di carattere prevalentemente compositivo, comparativo e di relazione:

1. *Lighting Conditions*: grafico di composizione a ciambella che mostra la ripartizione degli incidenti per condizione di luce. In particolare, si nota che la maggior parte degli incidenti stradali avviene in pieno giorno.



Figura 3.9: Pagina dell'applicativo APEX contenente la Dashboard 2 sulle condizioni ambientali e stradali

2. *Crash Severity by Critical Weather Conditions*: grafico di comparazione e relazione combinato; le barre mostrano il numero di incidenti per condizione meteo critica, mentre la linea sull'asse verticale secondario (Y2) rappresenta la percentuale di incidenti severi. La query sorgente raggruppa per `weather_group` e calcola il conteggio e la severità percentuale (Figura 3.10); nel calcolo della severità viene utilizzata una variabile booleana, `is_severe`, che vale 1 se l'incidente presenta feriti gravi o fatali, 0 altrimenti. Da notare come la pioggia concentri maggiori volumi di incidenti, mentre condizioni come nebbia o neve/ghiaccio, pur essendo meno frequenti, mostrano percentuali di severità più elevate.
3. *Crash Severity Type by Road Surface Condition*: grafico a barre orizzontali impilate 100% che definisce contemporaneamente una composizione e una comparazione; per ogni condizione della superficie stradale viene mostrata la ripartizione percentuale degli incidenti con feriti o traino del veicolo, rispetto a quelli senza feriti. Si osserva, ad esempio, che gli incidenti su strade bagnate (35% injuries/tow) o ghiacciate (31% injuries/tow) risultano più pericolosi, mentre quelli su strade innevate (24% injuries/tow) presentano meno feriti, nonostante la neve sia al secondo posto tra le condizioni meteorologiche critiche più frequenti, come mostrato nel grafico precedente.
4. *Crashes by Traffic Control Devices and their Conditions*: grafico a barre orizzontali impilate 100% a serie multipla. Anche qui si hanno una composizione e una comparazione; per ciascuna tipologia di dispositivo di controllo del traffico viene mostrata la percentuale di funzionanti, non presenti e non funzionanti. Dal grafico si osserva che i dispositivi funzionanti coprono la maggior parte dei casi, soprattutto i semafori; guardando, però,

```

1  with critical_weather_group as (
2    select weather_group,
3      case
4        when nvl(injuries_fatal,0) > 0 or nvl(injuries_incapacitating,0) > 0 then 1
5        else 0
6      end as is_severe
7      from a01_traffic_accidents_vw
8      where weather_group not in ('UNKNOWN/OTHER', 'CLEAR')
9  )
10 select weather_group,
11   count(*) as total_crashes,
12   round(100 * sum(is_severe) / count(*), 2) as severe_pct
13   from critical_weather_group
14 group by weather_group
15 order by total_crashes desc;

```

Figura 3.10: Query sorgente del grafico *Crash Severity by Critical Weather Conditions*

la somma delle quote di “no control” o “not functioning” sugli attraversamenti pedonali o scolastici (pedestrian/school: 39%) e in prossimità dei passaggi a livello (railroad: 21%), si evidenziano punti sensibili per la sicurezza urbana.

5. *Crashes by Lane Count and Top Road Defects*: grafico a barre verticali a serie multiple, di comparazione e relazione. Si confronta, per ciascun intervallo di numero di corsie (lane bucket), la ripartizione percentuale degli incidenti tra le tre principali categorie di difetti stradali (debris, surface damage e shoulder defect). La query SQL riportata in Figura 3.11 calcola, per ogni combinazione di corsie e difetto, il numero di incidenti e la relativa percentuale all’interno dello stesso gruppo di corsie, tramite la funzione analitica `SUM(crash_count) OVER (PARTITION BY lane_bucket)`. Dal grafico si nota, ad esempio, come i danni di superficie crescano all’aumentare delle corsie; in particolare, le strade ad una corsia riscontrano in percentuale più incidenti dovuti a detriti (debris: 7%) e a difetti della banchina stradale (shoulder defect: 12%), mentre per gli incidenti con erosione della superficie stradale, le più pericolose rimangono le strade a 5 o più corsie (surface damage: 88%).

```

1  with lanes as (
2    select lane_bucket,
3      road_defect_group,
4      count(*) as crash_count
5      from a01_traffic_accidents_vw
6      where lane_bucket not in ('UNKNOWN/INVALID')
7      and road_defect_group in ('DEBRIS', 'SURFACE DAMAGE', 'SHOULDER DEFECT')
8    group by lane_bucket, road_defect_group
9  )
10 select lane_bucket,
11   road_defect_group,
12   round(100 * crash_count / sum(crash_count) over (partition by lane_bucket), 2) as crash_count_pct
13   from lanes
14 order by lane_bucket, road_defect_group;

```

Figura 3.11: Query sorgente del grafico *Crashes by Lane Count and Top Road Defects*

Nel complesso, i risultati di queste analisi suggeriscono di ottimizzare la manutenzione stradale e la segnaletica nelle aree più esposte a pioggia o superfici danneggiate. Misure aggiuntive si possono proporre anche per potenziare i controlli e l’illuminazione nei punti critici, come gli attraversamenti pedonali e i passaggi a livello.

3.3.3 Dashboard 3: Cause e dinamiche

La terza dashboard analizza le principali cause e dinamiche degli incidenti, con l'obiettivo di mettere in relazione cause, tipologie di urto e velocità dei veicoli con i volumi e la severità degli eventi (Figura 3.12). I campi della vista utilizzati riguardano le cause primarie, il tipo di primo impatto, il numero di unità coinvolte e i limiti di velocità. Riguardo al campo delle cause primarie, sono state analizzate nel dettaglio le manovre improprie e le condizioni psico-fisiche del conducente, in modo da offrire considerazioni più specifiche su questi aspetti.

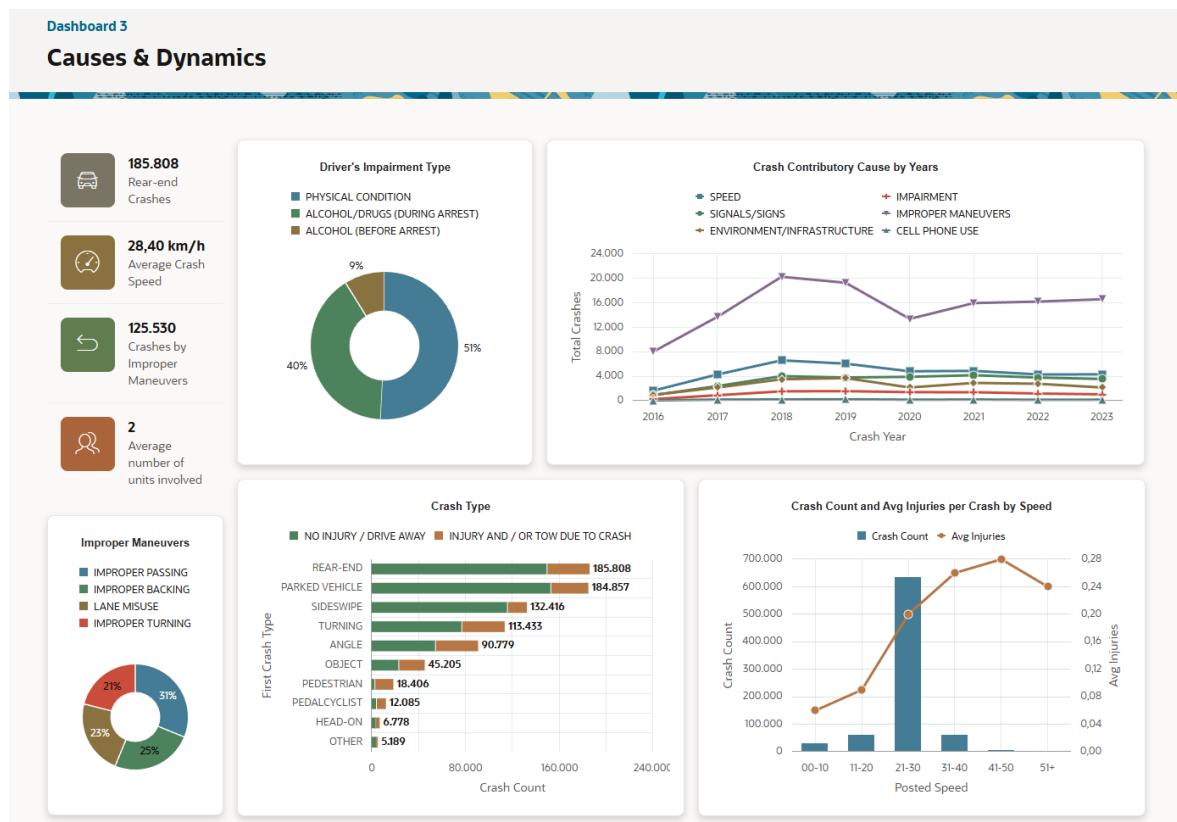


Figura 3.12: Pagina dell'applicativo APEX contenente la Dashboard 3 sulle cause e le dinamiche

I KPI calcolati sono i seguenti:

1. *Rear-end Crashes*: numero di tamponamenti, una delle dinamiche più comuni.
2. *Average Crash Speed*: limite di velocità media segnalato nei tratti interessati dagli incidenti.
3. *Crashes by Improper Maneuvers*: numero di incidenti dovuti principalmente a manovre improprie.
4. *Average Number of Units Involved*: numero medio di unità coinvolte per incidente.

Per questa dashboard sono stati realizzati cinque grafici, prevalentemente di composizione, comparazione e relazione:

1. *Driver's Impairment Type*: grafico a ciambella che rappresenta la composizione degli incidenti in base alla condizione psico-fisica del conducente. Le percentuali mostrano come l'assunzione di alcool sia uno dei principali stati di alterazione e, per la maggior parte dei casi, rilevata addirittura al momento stesso dell'arresto.

2. *Crash Contributory Cause by Years*: grafico a linee a serie multiple che confronta nel tempo il numero di incidenti attribuiti ad alcuni gruppi selezionati di cause contributive. Rientra nelle categorie di distribuzione e comparazione, poiché mostra l'andamento delle principali cause dal 2016 al 2023, con un picco tra il 2018 e il 2019 e un successivo calo, seguito da un leggero rialzo, a partire dal 2020. Le cause più rilevanti restano, nel tempo, sempre le manovre improvvise, il mancato rispetto dei limiti di velocità e le condizioni psico-fisiche del conducente.
3. *Improper Maneuvers*: grafico a ciambella che mostra la ripartizione percentuale degli incidenti attribuiti alle principali manovre improvvise. Si tratta di una composizione che permette di identificare rapidamente quali comportamenti scorretti contribuiscono maggiormente agli incidenti, come lo sono, principalmente, manovre errate di sorpasso e retromarcia.
4. *Crash Type*: grafico a barre orizzontali impilate a serie multiple che combina composizione e comparazione. Per ciascun tipo di primo impatto, viene mostrata la suddivisione tra incidenti senza feriti e incidenti con feriti e/o traino del veicolo. Il grafico conferma l'elevata frequenza dei tamponamenti e degli urti con veicoli parcheggiati, mettendo anche in evidenza quali tipologie presentano un maggior livello di gravità relativa, come gli incidenti con pedoni e ciclisti. L'immagine in Figura 3.13 mostra l'aggiornamento della visualizzazione del grafico che si ottiene cliccando singolarmente sui due campi della legenda interattiva.

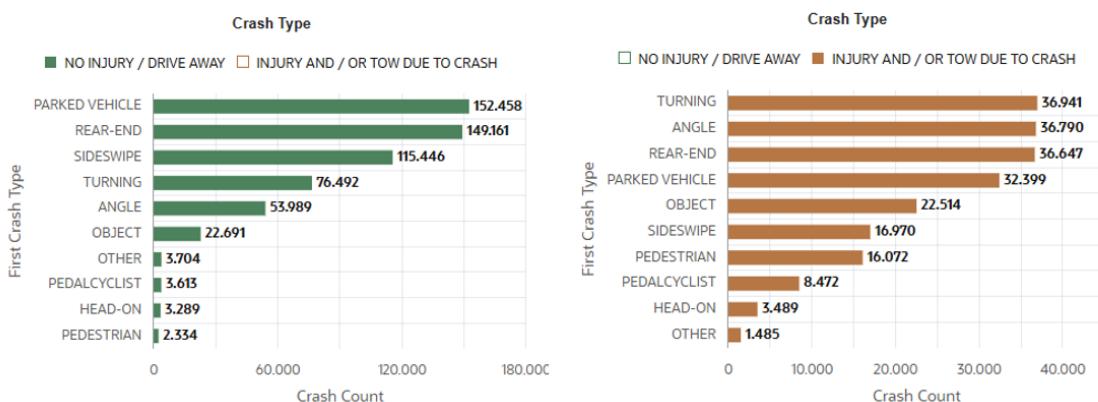


Figura 3.13: Visualizzazione dinamica dei dati nel grafico *Crash Type* tramite legenda interattiva

5. *Crash Count and Avg Injuries per Crash by Speed*: grafico combinato con barre verticali e linea su asse secondario, che mette in relazione il numero di incidenti e il numero medio di feriti per classi di limite di velocità (*speed_bucket*). Le barre rappresentano il conteggio degli incidenti per ciascun intervallo, mentre la linea indica la media delle persone ferite per incidente. Dal grafico emerge che gli incidenti si concentrano sui limiti intermedi, come 21–30 km/h e 31–40 km/h, mentre la media dei feriti tende a crescere all'aumentare del limite di velocità.

Tutte le osservazioni scaturite da questa analisi possono, ad esempio, servire a indirizzare campagne di sicurezza mirate contro le manovre pericolose e l'abuso di alcol, oltre che a rivedere i limiti di velocità e la segnaletica nei tratti dove si osserva un aumento della severità degli incidenti.

3.3.4 Dashboard 4: Analisi geografica

La quarta dashboard analizza la distribuzione spaziale degli incidenti, con particolare attenzione agli incroci, alle aree di cantiere e ai tratti stradali più critici (Figura 3.14). I campi della vista utilizzati comprendono le coordinate geografiche, la tipologia della carreggiata, la presenza di zone di lavoro, le informazioni sul distretto di polizia e i nomi delle strade con la più alta frequenza di incidenti.

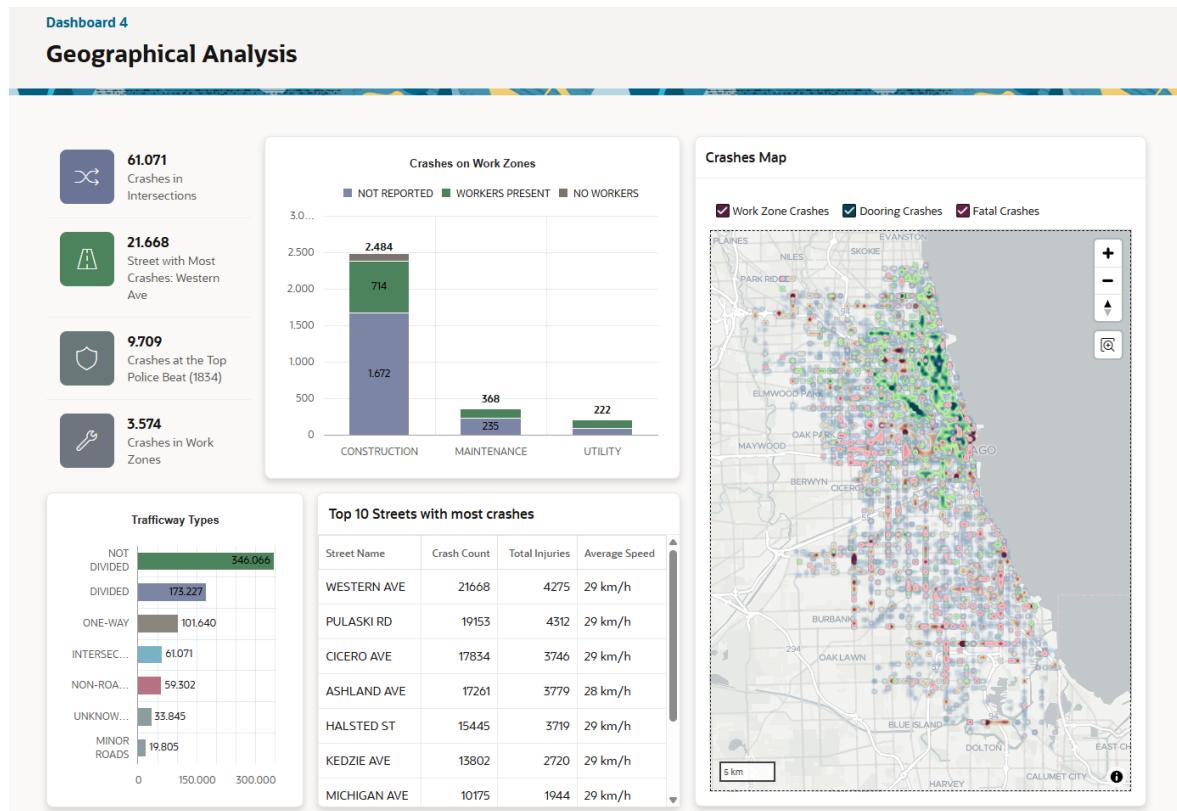


Figura 3.14: Pagina dell'applicativo APEX contenente la Dashboard 4 sull'analisi geografica

I KPI calcolati sono i seguenti:

1. *Crashes in Intersections*: numero di incidenti avvenuti in corrispondenza di incroci; evidenzia la criticità delle intersezioni come punti ad alta concentrazione di incidenti.
2. *Street with Most Crashes: Western Ave*: identifica la strada con il maggior numero di incidenti, utile per individuare una zona prioritaria di intervento.
3. *Crashes at the Top Police Beat (1834)*: numero di incidenti localizzati nel distretto di polizia con più casi; fornisce un riferimento operativo per le unità di polizia.
4. *Crashes in Work Zones*: numero di incidenti in aree di cantiere; misura il rischio associato alle attività svolte nei cantieri.

La sezione centrale della dashboard è formata da quattro componenti di natura descrittiva, comparativa e spaziale:

1. *Crashes on Work Zones*: grafico a barre verticali impilate che mostra, per ciascuna tipologia di cantiere, il numero di incidenti ripartito in base alla presenza o meno di lavoratori. Il grafico consente di confrontare le diverse categorie di cantiere ed evidenzia come i

volumi maggiori si concentrano maggiormente nelle aree di costruzione. In generale, l'elevata quota di incidenti con lavoratori presenti suggerisce l'esigenza di misure di protezione dedicate nei cantieri.

2. *Trafficway Types*: grafico di comparazione a barre orizzontali, che riassume il numero di incidenti per tipologia di carreggiata stradale. Si osserva che le strade a carreggiata non divisa rappresentano una porzione rilevante degli incidenti, seguite immediatamente da quelle a carreggiata divisa, poi da quelle a senso unico e, successivamente, dalle intersezioni.
3. *Top 10 Streets with most crashes*: report classico che elenca le dieci strade con il maggior numero di incidenti, riportando per ciascuna il conteggio totale (Crash Count), il numero complessivo di feriti (Total Injuries) e la velocità media limite (Average Speed). La tabella consente di individuare rapidamente le zone maggiormente critiche, come Western Ave, Pulaski Rd e Cicero Ave, e di valutarne la pericolosità.
4. *Crashes Map*: mappa cromatica interattiva che rappresenta la distribuzione spaziale degli incidenti tramite tre 'layer' selezionabili: Work Zone Crashes, Dooring Crashes e Fatal Crashes. Ogni layer è definito da una query che estrae le coordinate dalla vista A01_TRAFFIC_ACCIDENTS_VW, filtrando, rispettivamente, per incidenti su cantieri, incidenti da portiera e incidenti con la presenza di almeno un decesso. Nel Page Designer, i valori di LONGITUDE e LATITUDE vengono mappati come colonne della posizione da riportare nella mappa (Figura 3.15).

```

select latitude, longitude
from a01_traffic_accidents_vw
where latitude is not null and longitude is not null
and work_zone = 'Y'

select latitude, longitude
from a01_traffic_accidents_vw
where latitude is not null and longitude is not null
and dooring = 'Y'

select latitude, longitude
from a01_traffic_accidents_vw
where latitude is not null and longitude is not null
and nvl(injuries_fatal, 0) > 0
  
```

Figura 3.15: a) Struttura gerarchica dei componenti della *Crashes Map*, visualizzata nel riquadro sinistro del Page Designer). b) Mapping delle coordinate per la rappresentazione dei singoli incidenti nella mappa. c) Query sorgente per i tre layer

Queste informazioni possono essere utilizzate per pianificare in modo più efficace gli interventi di manutenzione e segnaletica, per migliorare la sicurezza nei lavori stradali e per ottimizzare la distribuzione delle pattuglie nei distretti più critici.

3.3.5 Dashboard 5: Esiti e severità

La quinta dashboard sintetizza gli esiti degli incidenti in termini di presenza di feriti, gravità delle lesioni, livelli di danno economico e quota di casi con fuga del conducente, con particolare attenzione alle variazioni nel tempo e alle fasce orarie più critiche (Figura 3.16). I campi della vista impiegati riguardano le tipologie di ferite e il loro conteggio, le classi di danno, l'indicatore di fuga e l'orario dell'incidente.

I KPI calcolati sono i seguenti:

1. *Crashes with Injuries*: numero di incidenti con almeno un ferito.
2. *Fatal Crashes*: numero di incidenti con almeno una vittima.

3. *Crashes with Damage over 500\$*: incidenti con danni economici rilevanti, oltre i 500\$.
4. *Hit and Run Crashes*: incidenti in cui almeno un conducente si è dato alla fuga.



Figura 3.16: Pagina dell'applicativo APEX contenente la Dashboard 5 su esiti e severità

I grafici realizzati per l'analisi sono i seguenti:

1. *Injuries by Police Report Type*: grafico a barre orizzontali impilate 100% che incrocia la tipologia di rapporto di polizia con il gruppo di severità di eventuali ferite. Si tratta di una composizione e comparazione che mostra, per ogni livello di gravità delle lesioni, la quota di incidenti rilevati sul campo rispetto a quelli registrati da remoto. Dal grafico emerge che gli incidenti più gravi sono quasi sempre associati a rapporti sul posto, mentre per i casi senza feriti aumenta il peso delle segnalazioni effettuate da remoto.
2. *Year Over Year Change by Most Severe Injury (Current vs Prev Year)*: grafico combinato a barre verticali e linea con area, che consente di analizzare, per un anno selezionato, la variazione del numero di incidenti rispetto all'anno precedente per ciascun gruppo di severità delle ferite. Le barre rappresentano la differenza assoluta (`delta_crash_count`), mentre la linea mostra la variazione percentuale (`delta_perc_crash_count`). La query sorgente utilizza una CTE, simile a quella del grafico in Figura 3.8, per calcolare il valore di `prev_crash_count` tramite funzione analitica `LAG(...)` `OVER (PARTITION BY most_severe_injury_group ORDER BY crash_year)`; dalla stessa CTE vengono poi derivati i delta assoluti e i delta percentuali. Il filtro sull'anno è gestito dall'elemento di pagina `P47_YEAR` (Figura 3.17), di tipo 'Selezione uno', che utilizza la lista di valori condivisa `A01_YEARS`; questo elemento è inserito tra gli 'Elementi pagina da sottomettere', così che la query del grafico venga ricalcolata ad

ogni variazione del valore selezionato nel menù a tendina dedicato. La visualizzazione permette di individuare in modo rapido gli anni in cui aumentano in modo significativo gli incidenti con esiti più gravi, come mostrato selezionando l'anno 2017.

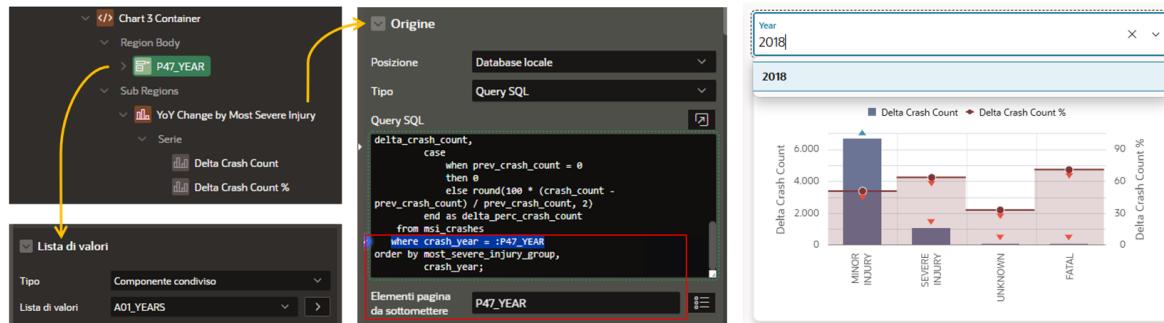


Figura 3.17: a) Struttura gerarchica del grafico *Year Over Year Change by Most Severe Injury* con al di sopra l'elemento di pagina P47_YEARS per il menù a tendina dinamico. b) Selezione della lista dei valori A01_YEARS. c) Visualizzazione del grafico e del menù a tendina subito dopo la selezione del valore '2018'; da notare la comparsa di piccoli indicatori triangolari, rossi e blu, per evidenziare incrementi o decrementi nei valori del grafico

3. *Crashes Damage Levels*: grafico di composizione a ciambella che mostra la ripartizione degli incidenti per classe di danno economico. La lettura evidenzia la quota non trascurabile di urti con danni elevati, superiori a \$1500, ed è utile per fare valutazioni assicurative e di impatto economico complessivo.
4. *No Injury Crashes* e *Fatal Crashes*: due grafici a torta che confrontano, rispettivamente per gli incidenti senza feriti e per quelli fatali, la percentuale di incidenti con fuga del conducente (hit & run) rispetto ai casi senza fuga (non hit & run). Entrambe le visualizzazioni sono di composizione e mostrano come gli episodi di fuga siano relativamente più frequenti tra gli incidenti più gravi.
5. *Injury Severity by Hour of the Day*: grafico combinato a linee con doppio asse verticale che rappresenta, per ciascuna ora del giorno, il numero di feriti fatali e il numero di feriti non fatali. Si tratta di una distribuzione temporale con comparazione tra le due serie. Il grafico evidenzia come il picco orario di incidenti non fatali sia compreso tra le 15:00 e le 16:00, mentre la presenza di fatalità è più frequente la sera, con picchi alle 20:00 e alle 23:00.

Tali informazioni risultano particolarmente rilevanti per le autorità di pubblica sicurezza e per gli enti assicurativi, al fine di ridurre la mortalità e i costi derivanti dagli incidenti più gravi.

CAPITOLO 4

Progettazione e implementazione dei grafici avanzati

In questo capitolo verrà affrontata la parte di analisi predittiva del progetto, con l'obiettivo di stimare l'andamento futuro del numero di incidenti stradali, a partire dai dati storici contenuti nel dataset. Dopo una breve introduzione sui concetti fondamentali del machine learning e delle serie storiche, l'analisi verrà formulata come un caso di regressione supervisionata sui dati temporali. Di conseguenza, l'ultima sezione del capitolo seguirà in dettaglio la costruzione del modello di previsione e dei grafici avanzati, implementati in un ambiente Python a partire dalla serie storica degli incidenti.

4.1 Introduzione al machine learning

Negli ultimi anni l'intelligenza artificiale (IA) è diventata una tecnologia chiave in molti settori, dalla finanza alla medicina, fino al marketing e alla sicurezza su più fronti, compresi i trasporti. In modo sintetico, si definisce intelligenza artificiale l'insieme di tecniche che permettono ai computer e alle macchine di simulare alcune capacità tipicamente umane, come l'apprendimento, la comprensione del linguaggio, la risoluzione di problemi e il processo decisionale. Da diverse fonti, comprese le pagine ufficiali dell'IBM e di Oracle, l'IA comprende tutte quelle soluzioni in cui un sistema è in grado di analizzare dati, riconoscere pattern e prendere decisioni con un certo grado di autonomia, migliorando nel tempo le proprie prestazioni man mano che viene esposto a nuove informazioni.

Tra le diverse aree specialistiche dell'intelligenza artificiale si evidenziano:

- *il machine learning (ML)*: si occupa di far apprendere ai sistemi dei modelli a partire dai dati;
- *l'elaborazione del linguaggio naturale*: permette ai sistemi di comprendere, interpretare e generare il linguaggio umano, e alimenta strumenti come gli assistenti vocali, i servizi di traduzione e i chatbot;
- *la computer vision*: consente l'interpretazione di immagini e video.

Nel contesto di questo progetto, l'attenzione è rivolta in particolare al machine learning, come strumento per analizzare e prevedere fenomeni nel tempo. A partire dalla serie storica degli incidenti stradali, infatti, si sfrutteranno dei modelli di previsione per stimare l'andamento futuro di una variabile di interesse come, ad esempio, il numero di incidenti in un certo intervallo di tempo.

4.1.1 Metodi di machine learning

Come appena accennato, il machine learning è una branca dell'intelligenza artificiale che utilizza metodi statistici per migliorare la performance di un algoritmo nell'identificare pattern nei dati. Si può considerare come una variante della programmazione tradizionale, nella quale in una macchina si predispone l'abilità di apprendere o dedurre qualcosa dai dati in maniera autonoma, senza ricevere istruzioni esplicite.

All'interno del machine learning si distinguono diverse famiglie di metodi, che differiscono per il tipo di problema affrontato e per il modo in cui i dati vengono utilizzati durante l'addestramento. Più nello specifico, a seconda del tipo di informazione e poi di "feedback" fornito al sistema, i compiti del ML vengono classificati nelle seguenti tre categorie, dette paradigmi (Figura 4.1):

1. *Apprendimento supervisionato* (supervised learning): all'algoritmo del modello vengono forniti in ingresso dei dati "etichettati", ossia degli esempi di possibili input con i rispettivi output desiderati, e l'obiettivo è quello di estrarre una regola generale che associa ogni nuovo input all'output corretto.
2. *Apprendimento non supervisionato* (unsupervised learning): il modello ha lo scopo di trovare dei pattern negli input forniti, senza che questi siano accompagnati da etichette o valori di output.
3. *Apprendimento per rinforzo*: l'algoritmo, chiamato agente, interagisce con un ambiente e impara tramite "trial and error"; ciò significa che ad ogni azione il modello riceve un certo segnale di ricompensa dall'ambiente, e l'obiettivo è trovare una strategia che massimizzi questo valore cumulato nel lungo periodo.

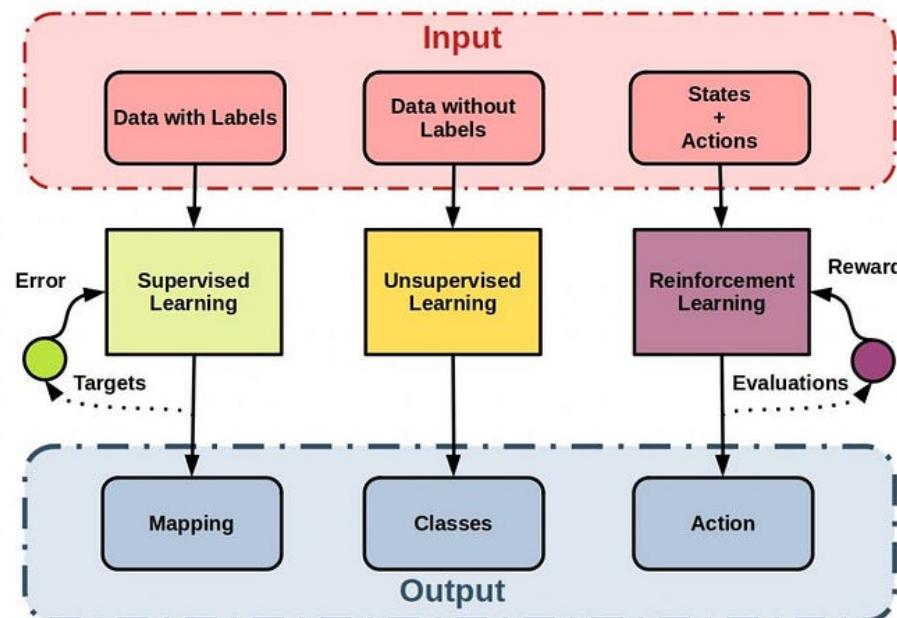


Figura 4.1: I tre principali paradigmi di apprendimento di un modello di machine learning

Nel caso dell'analisi avanzata sui dati degli incidenti stradali, ci si sofferma sull'apprendimento supervisionato, poiché è il paradigma più indicato per affrontare problemi di previsione su dati storici.

4.1.2 Supervised Learning

Nel metodo dell'apprendimento supervisionato, viene fornito al sistema un dataset etichettato con l'obiettivo di far apprendere la relazione tra i dati di input e le etichette di output. Il processo di addestramento prevede la regolazione iterativa dei parametri del modello per ridurre al minimo la differenza tra gli output predetti e quelli effettivi dei dati di test. Una volta addestrato, il modello può prevedere gli output corretti di nuovi dati non etichettati.

Alcuni esempi comuni di input e output etichettati includono l'associazione di una foto di un animale al suo nome (Figura 4.2) oppure la distinzione di un'email tra le categorie "spam" o "non spam".

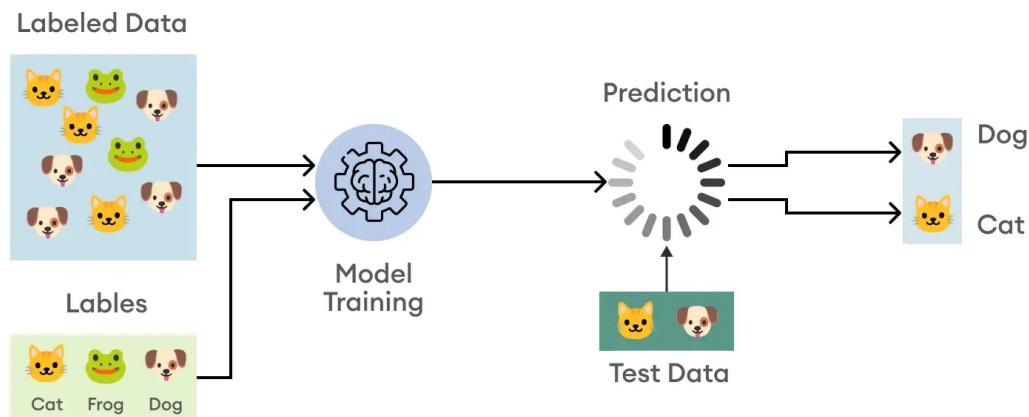


Figura 4.2: Esempio di algoritmo di classificazione di supervised learning

Nel supervised learning, un problema viene formulato partendo da un dataset costituito da n osservazioni:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

dove:

- x_i è il vettore di input (o *feature*) che descrive la i -esima osservazione;
- y_i è l'output (o *target*) associato a tale osservazione.

L'obiettivo è stimare una funzione f tale che:

$$y \approx f(x)$$

cioè una funzione che, dato un nuovo vettore di input x , fornisca una buona approssimazione del corrispondente valore di output y .

Secondo il testo "*The Elements of Statistical Learning*" di Hastie, Tibshirani e Friedman, la formulazione tipica di un problema di supervised learning è:

$$Y = f(X) + \varepsilon$$

dove ε rappresenta il rumore, cioè la componente casuale, non descrivibile in modo deterministico dal modello.

A seconda della natura della variabile Y , si distinguono due classi principali di problemi di supervised learning (Figura 4.3):

- *Classificazione*: il target Y è una variabile qualitativa categoriale, che assume un numero finito di classi. Il compito del modello è assegnare ciascuna osservazione alla categoria corretta. Nel contesto degli incidenti stradali, si potrebbe classificare un incidente come "grave" o "non grave", oppure associarlo ad una certa categoria di rischio.
- *Regressione*: il target Y è una variabile continua, quindi il modello deve prevedere un valore reale. Un esempio potrebbe essere la stima del numero di incidenti attesi in un certo intervallo temporale, oppure la previsione della gravità media o del numero di feriti associati a particolari condizioni ambientali o dinamiche del traffico.

Questi concetti verranno illustrati successivamente, con riferimento alla serie storica degli incidenti stradali che sarà trattata come un problema di regressione sui dati temporali.

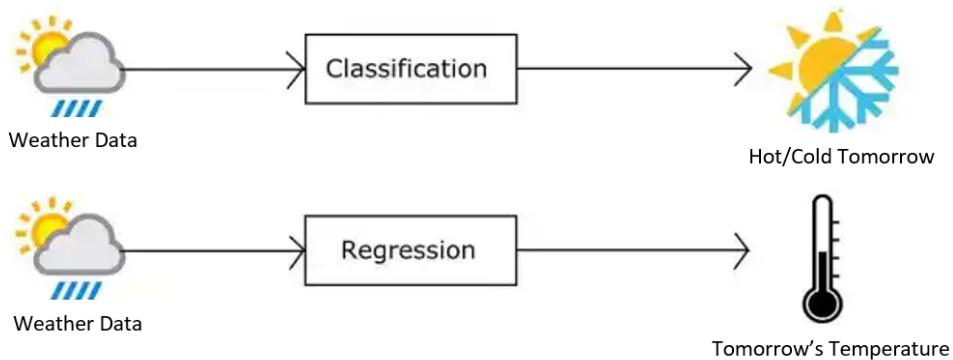


Figura 4.3: Esempi di classificazione e di regressione

4.2 Serie storiche

Una serie storica, o serie temporale, è una sequenza di osservazioni ordinate e registrate nel tempo a intervalli regolari. Nell'attuale caso d'interesse, alcuni esempi di serie storiche possono essere il numero di incidenti giornalieri, settimanali o mensili.

4.2.1 Componenti principali

Lo scopo dell'analisi delle serie storiche consiste nello studio dell'evoluzione passata del fenomeno rispetto al tempo, mentre la previsione si ottiene ipotizzando che tali regolarità di comportamento si ripetano nel futuro. Le informazioni di una serie storica possono essere analizzate prendendo in esame le singole componenti che la caratterizzano e che aiutano a identificare pattern, tendenze e irregolarità nei suoi dati.

Le principali componenti di una serie storica sono (Figura 4.4):

- *trend*: andamento a lungo termine dei dati, che può essere crescente o decrescente e che riflette l'evoluzione media del fenomeno nel tempo;
- *stagionalità*: schema che si ripete nei dati a intervalli regolari, come nel caso delle variazioni annuali;
- *ciclicità*: schema che si ripete nei dati dopo un certo numero di osservazioni e che non è necessariamente correlato alla stagionalità;
- *irregolarità e rumore*: componenti che non sono spiegate dal trend o dalla stagionalità, ma sono dovute a fluttuazioni imprevedibili e casuali nei dati della serie.

Identificando bene queste componenti, è possibile comprendere meglio la struttura dei dati che si hanno a disposizione e formulare analisi più accurate.

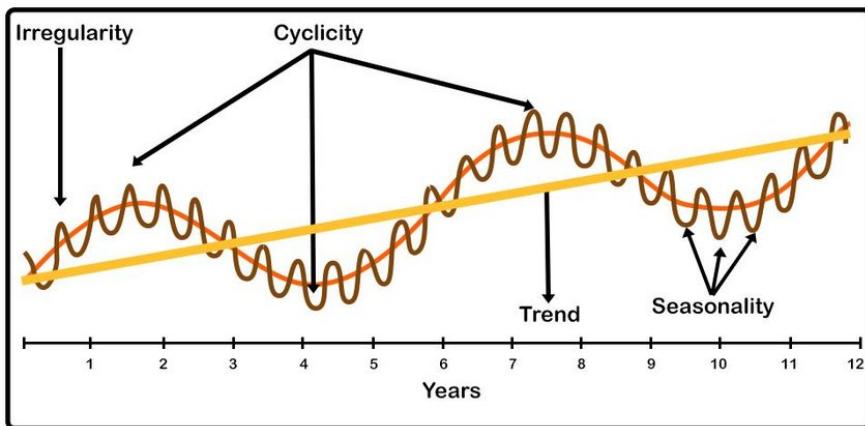


Figura 4.4: Rappresentazione grafica della scomposizione di una serie storica nelle sue quattro componenti principali

4.2.2 Analisi descrittiva e analisi predittiva

Dopo l'introduzione alle analitiche nel Capitolo 3, si può ora trattare la natura delle analisi utilizzate nell'ambito delle serie temporali. In questo contesto si distinguono le seguenti analisi principali:

- *Analisi descrittiva (time series analysis)*: implica lo sviluppo di modelli che descrivono la serie temporale osservata, al fine di comprenderne le cause sottostanti e interpretare al meglio le informazioni presenti nei dati. Questo tipo di analisi si concentra, quindi, sulla comprensione della struttura della serie e sull'individuazione delle sue componenti principali.
- *Analisi predittiva (time series forecasting)*: consiste nell'utilizzo di modelli basati sui dati della serie per prevedere osservazioni e valori futuri.

L'analisi descrittiva è quindi adatta alla fase esplorativa dei dati, che in questo progetto è svolta dalle cinque dashboard presenti nell'applicativo APEX. L'analisi predittiva, invece, mira a stimare l'evoluzione futura del numero di incidenti; di conseguenza, nelle prossime sezioni verranno documentate la trasformazione della serie storica degli incidenti in un problema di regressione supervisionata e l'applicazione di un modello di machine learning implementato in Python.

4.3 Addestramento e valutazione del modello

Nel caso della serie storica degli incidenti stradali, la previsione dell'andamento futuro di una variabile, come, ad esempio, il numero di incidenti in un certo intervallo di tempo, può essere vista come un problema di regressione supervisionata. In questo contesto il target è il valore della serie in un certo istante futuro, mentre i dati di input possono includere sia informazioni sul passato della serie sia eventuali variabili esplicative.

In questa sezione verranno descritti il processo di suddivisione dei dati e le metriche utilizzate per valutare la qualità del modello applicato alla serie storica.

4.3.1 Suddivisione temporale dei dati

Nella maggior parte delle attività di supervised learning, le best practice consigliano di suddividere i dati in tre set indipendenti, al fine di valutare in modo realistico le prestazioni del modello:

- *Set di addestramento (training set)*: è il set utilizzato dal modello per apprendere pattern e relazioni utili alle previsioni da formulare in seguito. Questo set deve essere il più rappresentativo e imparziale possibile, poiché qualsiasi distorsione in questa fase potrebbe ripercuotersi sulle previsioni.
- *Set di convalida (validation set)*: è il set utilizzato per confrontare le prestazioni di modelli diversi e scegliere quello che fornisce una migliore generalizzazione.
- *Set di test (test set)*: è il set utilizzato per la valutazione finale delle prestazioni del modello. È importante che esso venga impiegato solo alla fine, dopo l'identificazione del modello migliore, in modo da valutarlo in maniera imparziale.

A differenza di altri contesti di machine learning, nelle serie temporali questa suddivisione del dataset non avviene mescolando casualmente i dati, ma facendo un taglio temporale. In questo modo, si va ad addestrare il modello con i dati del passato e lo si testa con quelli del futuro, così da evitare la visione anticipata di informazioni future.

Nel caso di questo progetto, si va a sfruttare direttamente un certo modello di regressione già esistente, per cui i dati storici sugli incidenti sono stati suddivisi esclusivamente nei due set di training (anni 2016-2021) e di test (anni 2022-2023), senza la necessità del set di convalida.

4.3.2 Metriche di errore

Una volta calcolate le previsioni sul test set, è necessario valutare l'accuratezza del modello tramite delle metriche di errore. Nei problemi di regressione vengono solitamente utilizzate le seguenti metriche:

- *errore assoluto medio (MAE)*: la media dei valori assoluti degli errori;
- *scarto quadratico medio (RMSE)*: la radice quadrata della media dei quadrati degli errori;
- *errore percentuale assoluto medio (MAPE)*: l'errore in termini percentuali rispetto al valore osservato.

In particolare, il MAE e l'RMSE sono metriche dipendenti dalla scala della variabile e forniscono indicazioni complementari. Grazie all'assenza dell'elevazione al quadrato, il MAE è più robusto ai valori anomali e ai grandi errori, mentre l'RMSE li penalizza maggiormente e risulta quindi utile nella valutazione di scostamenti critici.

4.4 Realizzazione dei grafici avanzati con Python

In aggiunta alle dashboard descrittive sviluppate in Oracle APEX, la parte avanzata del progetto prevede la costruzione di un modello di previsione del numero di incidenti nel tempo e la visualizzazione grafica dei valori osservati e delle previsioni.

Inizialmente si era previsto di realizzare la componente di analisi avanzata in Oracle Machine Learning for Python (OML4Py), tramite notebook eseguiti su Oracle Autonomous

Database. Tuttavia, durante le prove l’ambiente cloud ha smesso di funzionare in modo affidabile, presumibilmente a causa delle limitazioni di risorse del workspace gratuito nell’utilizzo di un dataset di circa 700.000 record. Di conseguenza, si è scelto di spostare il processo di analisi in un ambiente Python locale, basato su JupyterLab, DuckDB e scikit-learn. Il cambio di tecnologia non ha modificato in alcun modo l’impostazione metodologica dell’analisi, ma ha reso l’esecuzione più controllabile e indipendente dall’infrastruttura cloud.

4.4.1 Ambiente di analisi e strumenti

L’ambiente di analisi è stato configurato creando un ambiente virtuale Python in una cartella di lavoro locale. I comandi eseguiti dal terminale sono mostrati nella Figura 4.5.

```
PS C:\JupyterLabNBs> python -m venv venv1
PS C:\JupyterLabNBs> .\venv1\Scripts\activate
(venv1) PS C:\JupyterLabNBs> python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\jupyterlabnbs\venv1\lib\site-packages (25.1.1)
Collecting pip
  Using cached pip-25.3-py3-none-any.whl.metadata (4.7 kB)
  Using cached pip-25.3-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 25.1.1
    Uninstalling pip-25.1.1:
      Successfully uninstalled pip-25.1.1
Successfully installed pip-25.3
(venv1) PS C:\JupyterLabNBs> pip install -r requirements.txt
Collecting polars (from -r requirements.txt (line 1))
  Using cached polars-1.35.2-py3-none-any.whl.metadata (10 kB)
Collecting pandas (from -r requirements.txt (line 2))
  Using cached pandas-2.3.3-cp313-cp313-win_amd64.whl.metadata (19 kB)
Collecting pyodbc (from -r requirements.txt (line 3))
```

Figura 4.5: Comandi eseguiti da terminale per la creazione in locale dell’ambiente virtuale Python e per l’installazione delle librerie necessarie

Nel file `requirements.txt` sono state inserite le librerie necessarie all’esecuzione del notebook. Di seguito vengono elencate le principali:

- *jupyterlab*: per utilizzare l’ambiente notebook;
- *duckdb*: impiegato come database locale per caricare e interrogare il dataset;
- *pandas*: per la manipolazione dei dati tabellari, tramite degli oggetti chiamati dataframe;
- *numpy*: per le operazioni numeriche, come il calcolo delle metriche d’errore;
- *matplotlib*: per la costruzione dei grafici;
- *scikit-learn*: per l’importazione e l’addestramento del modello di machine learning utilizzato per le previsioni.

La scelta di un notebook Jupyter è stata fatta considerando la possibilità di combinare in un unico documento del codice eseguibile, del testo descrittivo e dei grafici, rendendo agevole la documentazione di tutto il lavoro. Nella Figura 4.6 compaiono i primi blocchi di script Python che eseguono rispettivamente queste operazioni:

1. Import delle librerie e del modello di machine learning, il Random Forest Regressor.
2. Connessione in-memory a un database DuckDB, per salvare il dataset in una tabella temporanea.

3. Caricamento, nella tabella temporanea `crashes_data`, dell'intero file CSV originale del dataset, dopo aver identificato automaticamente la sua struttura e i tipi di dato tramite il comando SQL `read_csv_auto`.
4. Conversione in un dataframe Pandas (`df`) dell'oggetto DuckDB restituito dalla SELECT sull'intera tabella `crashes_data`.
5. Visualizzazione dei primi cinque record tramite `df.head()`.

```
# 1) Setup: librerie, connessione e lettura dati

import duckdb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

con = duckdb.connect(database=':memory:') # connessione in-memory a un database DuckDB

# caricamento del CSV del dataset in una tabella DuckDB
con.sql("""
    CREATE TABLE crashes_data AS
    SELECT * FROM read_csv_auto('Kaggle_traffic_accidents_dataset.csv')
""")

# visualizzazione dei primi record come DataFrame Pandas
df = con.sql("SELECT * FROM crashes_data").df()
df.head() # mostra le prime 5 righe
```

	CRASH_RECORD_ID	CRASH_DATE_EST_I	CRASH_DATE	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE	DEVICE_CONDITION	WEATHER_CONDITION
0	23a79931ef555d54118f64dc9be2cf2dbf59636ce253f7...	None	2023-09-05 19:05:00	30	TRAFFIC SIGNAL	FUNCTIONING PROPERLY	CLEAR
1	2675c13fd0f474d730a5b780968b3cafc7c12d7adb661f...	None	2023-09-22 18:45:00	50	NO CONTROLS	NO CONTROLS	CLEAR

Figura 4.6: Script Python per il setup dell'ambiente, la connessione al database e la lettura del dataset

Dopo il salvataggio dell'intero dataset in un dataframe Pandas, è ora possibile utilizzare e manipolare tutti i dati tramite le funzioni messe a disposizione dalla libreria. Nella prossima sezione compariranno vari esempi a riguardo, come la conversione delle date in formati appropriati, l'ordinamento e il filtraggio dei record e il campionamento settimanale degli incidenti.

4.4.2 Preparazione dei dati e costruzione della serie storica settimanale

Dopo aver caricato l'intero dataset in un dataframe Pandas, è possibile procedere alla realizzazione della serie storica. Di seguito verranno documentati i seguenti passi:

- preparazione dei dati;
- costruzione della serie storica aggregata;
- suddivisione temporale della serie per il train e il test del modello di machine learning.

Partendo dal processo di preparazione dei dati degli incidenti, la Figura 4.7 mostra le operazioni che lo compongono. Si comincia dalla conversione della colonna con la data dell'incidente; la variabile `CRASH_DATE` viene trasformata in un oggetto `datetime` tramite la funzione `to_datetime()`, così da poter effettuare correttamente le operazioni temporali. L'opzione `errors = 'coerce'` converte automaticamente le date non valide in valori `NaT` (Not a Time) da scartare. Successivamente avviene la rimozione delle righe con una data non valida, attraverso la funzione `dropna()`. Infine, si ordinano temporalmente tutti i record, in modo da avere un dataframe Pandas cronologicamente coerente e pronto per le operazioni successive.

```
# 2) Preparazione dati: estrapolazione, pulizia e ordinamento

# Conversione della colonna con la data dell'incidente
df['CRASH_DATE'] = pd.to_datetime(df['CRASH_DATE'], errors='coerce') # converte la colonna CRASH_DATE in un oggetto data/ora

# toglie le righe con data non valida, come le NaT (Not a Time) generate dell'opzione di sicurezza 'coerce'
df = df.dropna(subset=['CRASH_DATE'])

df = df.sort_values('CRASH_DATE')
df.head()
```

	CRASH_RECORD_ID	CRASH_DATE_EST_I	CRASH_DATE	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE
535086	a802658be15312809c771559e4f81088cfb226830792a5...		None 2013-03-03 16:48:00	30	TRAFFIC SIGNAL

Figura 4.7: Script Python per l'estrapolazione, la conversione e l'ordinamento dei dati

Si può ora passare alla costruzione della serie storica. La scelta di una granularità settimanale ricade sull'intenzione di ridurre l'elevata variabilità giornaliera degli incidenti, legata a fattori come meteo, traffico o eventi occasionali. In questo modo, è più facile individuare tendenze e ciclicità, rendendo la serie più regolare e adatta ai modelli di forecasting.

Come mostrato in Figura 4.8, la colonna CRASH_DATE viene impostata come indice temporale del dataframe e utilizzata per creare un'aggregazione settimanale tramite la funzione Pandas `resample('W')`. Per ogni intervallo di una settimana viene calcolato il numero totale di incidenti (`num_crashes`) tramite il metodo `size()`, ottenendo così una serie temporale che rappresenta il volume settimanale degli eventi. Eventuali settimane prive di incidenti vengono compilate con il valore 0 attraverso la funzione `fillna(0)`. Come mostrato dall'output del comando `ts.head()`, il risultato di tale script è un dataframe (`ts`) che contiene, per ciascuna settimana, il numero di incidenti registrati nel dataset.

```
# 3) Costruzione della serie storica aggregata

# Granularità: settimanale
ts = (
    df.set_index('CRASH_DATE') # rende CRASH_DATE la colonna indice del DataFrame
    .resample('W') # raggruppa i dati secondo una freq. temporale ('W' = Weekly, con la domenica come termine di default)
    .size() # conta quanti incidenti ci sono in ciascun raggruppamento settimanale
    .rename('num_crashes') # rinomina la colonna del conteggio
    .to_frame() # riconversione in un DataFrame della singola colonna restituita da .size()
)

# riempie eventuali settimane senza incidenti con 0
ts['num_crashes'] = ts['num_crashes'].fillna(0)

ts.head()
```

	num_crashes
CRASH_DATE	
2013-03-03	1
2013-03-10	0
2013-03-17	0
2013-03-24	0
2013-03-31	0

Figura 4.8: Script Python per la costruzione della serie storica aggragata settimanalmente

In accordo con quanto accennato in precedenza, la serie storica verrà suddivisa temporalmente in:

- *un training set*: settimane dal 2016 alla fine del 2021, utilizzate per l'addestramento del modello;

- un *test set*: settimane dal 2022 alla fine del 2023, utilizzate per simulare le osservazioni future e valutare le previsioni del modello.

Dopo aver valutato gli anni che vanno dal 2016 alla fine del 2023 come un periodo sufficientemente ampio e omogeneo per l’analisi, la serie aggregata viene limitata a questo intervallo temporale, come mostrato in Figura 4.9. Successivamente, tramite l’istruzione `loc`, vengono generati i dataframe `train_ts` e `test_ts` per verificare la correttezza della suddivisione temporale. Gli output di `train_ts.head()` e `test_ts.head()` mostrano rispettivamente le prime cinque settimane del periodo di addestramento e del periodo di test, confermando che il taglio temporale è stato applicato correttamente e che ciascun sottoinsieme contiene una sequenza valida e ordinata.

```
# 4) Split temporale per il Train/Test

# Selezione della finestra temporale di lavoro
ts = ts.loc['2016-01-01':'2023-12-31'] # .Loc seleziona un sottoinsieme della serie storica

# Train: 2016-2021, Test: 2022-2023
split_date = '2022-01-01'

train_ts = ts.loc[:'2021-12-31']
test_ts = ts.loc[split_date:]
```

train_ts.head()		test_ts.head()	
		num_crashes	
		CRASH_DATE	
		2016-01-03	505
		2016-01-10	621
		2016-01-17	666
		2016-01-24	664
		2016-01-31	598
		2022-01-02	1615
		2022-01-09	1778
		2022-01-16	1806
		2022-01-23	1826
		2022-01-30	2039

Figura 4.9: Scelta della finestra di lavoro e suddivisione temporale della serie

Il corretto esito di questa procedura permette di applicare la stessa suddivisione anche al dataset supervisionato, che, come illustrato nella sezione successiva, sarà il dataset effettivo da impiegare nel modello di machine learning.

4.4.3 Problema supervisionato e modello di regressione

Per applicare un algoritmo di apprendimento supervisionato alla serie storica degli incidenti, questa è stata trasformata in un dataset di regressione tramite il metodo delle *lag feature*. Le lag feature sono semplicemente i valori passati della serie stessa, usati come variabili di input per prevedere il valore futuro. L’idea è quindi quella di costruire, per ogni settimana, un vettore di input che comprenda i conteggi degli incidenti nelle settimane precedenti, in modo da stimare quello della settimana corrente. Questi input “sfasati” vengono generati nel codice usando il metodo `shift()` di Pandas, che sposta la colonna dei valori osservati nel passato e permette di creare in modo compatto le *colonne di lag*.

Il codice per attuare questo processo può essere suddiviso nei seguenti passi:

- creazione del dataset supervisionato contenente le lag feature e le variabili di calendario;
- suddivisione del dataset supervisionato nei set di training e di test;
- scelta e configurazione del modello di regressione da allenare per le previsioni.

Per prima cosa si ricavano dalla serie le colonne di lag e le variabili di calendario, che serviranno da input per il modello. Nel codice in Figura 4.10 si può osservare la struttura della funzione `make_supervised()`, necessaria alla trasformazione della serie in un dataset adatto all'apprendimento supervisionato.

```
# 5) Trasformazione della serie in un problema supervisionato con lag features

# Funzione helper per creare le features di lag e di calendario
def make_supervised(serie, n_lags=7, freq='W'):

    # TARGET (output): colonna 'y' dei valori da prevedere
    df_sup = pd.DataFrame({'y': serie}) # nuovo dataframe per la colonna target

    # FEATURES (input): colonne 'x' degli input da dare al modello durante l'addestramento
    # Lag features
    for lag in range(1, n_lags + 1):
        df_sup[f'lag_{lag}'] = df_sup['y'].shift(lag)

    # Features di calendario
    idx = df_sup.index
    df_sup['month'] = idx.month
    df_sup['year'] = idx.year

    df_sup = df_sup.dropna()
    return df_sup
```

Figura 4.10: Trasformazione della serie in un problema supervisionato con lag feature

Questa funzione genera un dataframe, `supervised` (Figura 4.11) contenente rispettivamente le seguenti colonne:

- *target y*: il numero di incidenti nella settimana da prevedere;
- *sette lag feature* (`lag_1`, ..., `lag_7`): il numero di incidenti nelle sette settimane precedenti;
- *due feature di calendario*: il mese (`month`) e l'anno (`year`) della settimana in questione, per catturare eventuali effetti stagionali e di trend.

	supervised = make_supervised(ts['num_crashes'], n_lags=7, freq='W') # applica la funzione alla serie storica supervised.head()									
	y	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	month	year
CRASH_DATE										
2016-02-21	620	672.0	546.0	598.0	664.0	666.0	621.0	505.0	2	2016
2016-02-28	618	620.0	672.0	546.0	598.0	664.0	666.0	621.0	2	2016
2016-03-06	661	618.0	620.0	672.0	546.0	598.0	664.0	666.0	3	2016
2016-03-13	659	661.0	618.0	620.0	672.0	546.0	598.0	664.0	3	2016
2016-03-20	666	659.0	661.0	618.0	620.0	672.0	546.0	598.0	3	2016

Figura 4.11: Stampa del dataframe contenente il dataset supervisionato

In particolare, le colonne delle lag feature si ottengono tramite la funzione `shift(lag)` di Pandas che, utilizzata all'interno del ciclo `for`, sposta la colonna dei valori della serie

nel passato per un certo numero di passi (`n_lags = 7`). Le prime righe con valori mancanti, generate dallo `shift()` delle lag, vengono successivamente eliminate tramite `dropna()`, ottenendo così un dataset supervisionato privo di valori nulli.

Il dataframe supervisionato viene poi suddiviso nei due sottoinsiemi `train_sup` e `test_sup`, utilizzando lo stesso istante di taglio definito precedentemente (inizio 2022). Dal sottoinsieme di training vengono ricavati la matrice delle feature `X_train`, composta da tutte le colonne tranne quella degli output `y`, e il vettore target `y_train`; analogamente, dal sottoinsieme di test si ottengono `X_test` e `y_test` (Figura 4.12).

```
# Esegue lo split train/test sul nuovo dataframe supervisionato
train_sup = supervised.loc[supervised.index < split_date]
test_sup = supervised.loc[supervised.index >= split_date]

# Matrice delle Features (input) per l'addestramento (tutto train_sup tranne la colonna y)
X_train = train_sup.drop(columns=['y'])

# Vettore di Target (output) per l'addestramento (solo la colonna y)
y_train = train_sup['y']

# Stesse operazioni, ma per i set di test
X_test = test_sup.drop(columns=['y'])
y_test = test_sup['y']
```

X_train.head()	X_test.head()
CRASH_DATE	CRASH_DATE
2016-02-21 672.0	2022-01-02 1667.0
2016-02-28 620.0	2022-01-09 1615.0
2016-03-06 618.0	2022-01-16 1778.0
2016-03-13 661.0	2022-01-23 1806.0
2016-03-20 659.0	2022-01-30 1826.0
546.0 598.0 664.0 666.0 621.0 505.0	1985.0 2120.0 2090.0 1862.0 2139.0 2095.0
620.0 567.0 546.0 598.0 664.0 666.0	1667.0 1667.0 1985.0 2120.0 2090.0 1862.0
621.0	2139.0
2 2016	1 2022
618.0 620.0 672.0 546.0 598.0 664.0	1615.0 1667.0 1985.0 2120.0 2090.0 1862.0
620.0 672.0 546.0 598.0 664.0	1 2022
569.0 618.0 620.0 672.0 546.0 598.0	1778.0 1615.0 1667.0 1985.0 2120.0 2090.0
664.0	1 2022
598.0 664.0	1806.0 1778.0 1615.0 1667.0 1985.0 2120.0
666.0	1 2022
month year	month year
2016-03-20 659.0 661.0 618.0 620.0 672.0 546.0 598.0	2022-01-30 1826.0 1806.0 1778.0 1615.0 1667.0 1985.0 2120.0
3 2016	1 2022

y_train.head()	y_test.head()
CRASH_DATE	CRASH_DATE
2016-02-21 620	2022-01-02 1615
2016-02-28 618	2022-01-09 1778
2016-03-06 661	2022-01-16 1806
2016-03-13 659	2022-01-23 1826
2016-03-20 666	2022-01-30 2039
Freq: W-SUN, Name: y, dtype: int64	Freq: W-SUN, Name: y, dtype: int64

Figura 4.12: Taglio temporale del dataset supervisionato e creazione delle matrici degli input e dei vettori degli output, da fornire al modello durante il training e il testing

Come ultimo step di questo processo, si passa alla creazione e alla configurazione del modello di machine learning scelto per la regressione. In questo caso è stato utilizzato il *Random Forest Regressor* della libreria scikit-learn.

Per fornire un breve contesto teorico, un *Random Forest*, o foresta casuale, è un algoritmo di apprendimento automatico che combina l'output di più alberi decisionali per raggiungere un unico risultato. Nel machine learning, un *albero decisionale* è un modello predittivo con struttura gerarchica ad albero, composta da un nodo radice, da nodi interni, collegati da rami, e da nodi foglia (Figura 4.13).

Nei problemi di regressione, un albero decisionale possiede le seguenti caratteristiche:

- ogni nodo interno contiene una regola di suddivisione basata su una variabile di input (per esempio: "se $lag_1 \leq 25$ ");
- ogni ramo verso un nodo figlio rappresenta l'esito di quella regola (per esempio: il ramo sinistro corrisponde a " $lag_1 \leq 25$ ", mentre il ramo destro a " $lag_1 > 25$ ");
- ogni nodo foglia contiene il valore numerico predetto per la variabile obiettivo.

Come accenato anche di seguito, il valore in un nodo foglia corrisponde solitamente alla media dei valori della variabile obiettivo per tutti i record del training set che soddisfano le condizioni lungo il percorso dal nodo radice a quella foglia.

Di conseguenza, l'albero decisionale approssima complessivamente la funzione che collega gli input al valore da prevedere in un problema di regressione.

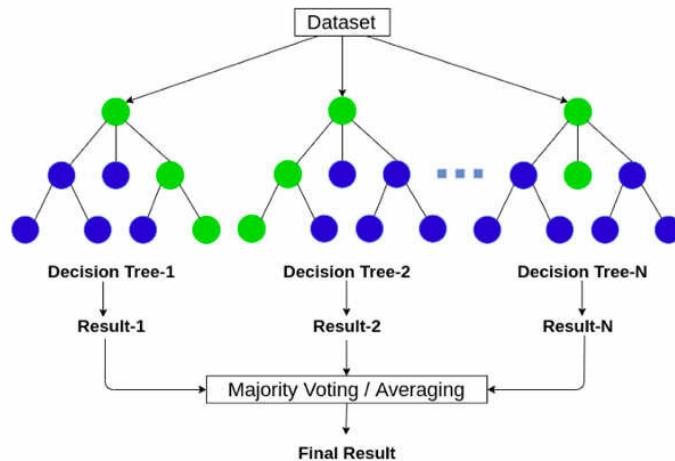


Figura 4.13: Esempio grafico di un algoritmo Random Forest in un problema di regressione

In particolare, il Random Forest Regressor è un metodo di apprendimento *d'insieme*. Ciò significa che invece di addestrare un solo albero decisionale, il modello ne costruisce molti, ognuno su un campione leggermente diverso dei dati di addestramento. Le previsioni finali vengono poi ottenute facendo la media delle previsioni dei singoli alberi. Inoltre, a ogni split di ciascun albero viene considerato solo un sottoinsieme casuale degli input disponibili; questa casualità sulle feature riduce la correlazione tra gli alberi e contribuisce a migliorare la capacità di generalizzazione del modello rispetto a quella di un singolo albero decisionale.

La Figura 4.14 mostra la creazione e la configurazione del modello di regressione scelto, un `RandomForestRegressor` della libreria `scikit-learn`.

```

# 6) Creazione modello, Addestramento e Previsione

rf = RandomForestRegressor(
    n_estimators=300, # numero di alberi decisionali
    random_state=42, # seme casuale per la riproducibilità imparziale dell'analisi
    n_jobs=-1 # permette l'uso di tutti i core della CPU per addestrare i 300 alberi in parallelo
)

rf.fit(X_train, y_train) # Training
y_pred = rf.predict(X_test) # Testing e calcolo delle previsioni
  
```

```

df_pred = pd.DataFrame({
    'y_actual': y_test.values,
    'y_pred': y_pred
}, index=y_test.index)

df_pred.head()
  
```

	y_actual	y_pred
CRASH_DATE		
2022-01-02	1615	1793.746667
2022-01-09	1778	1721.363333
2022-01-16	1806	1886.663333
2022-01-23	1826	1943.440000
2022-01-30	2039	1922.643333

Figura 4.14: Creazione e addestramento di un Random Forest Regressor per l'analisi predittiva

Il modello viene istanziato con 300 alberi decisionali (`n_estimators = 300`), un seed casuale fissato (`random_state = 42`), per garantire la riproducibilità dei risultati, e l'opzione `n_jobs = -1` per sfruttare tutti i core della CPU durante l'addestramento.

Il comando `fit(X_train, y_train)` avvia la fase di training, durante la quale il modello apprende le relazioni tra le lag feature, il mese e l'anno (matrice degli input) e il numero di incidenti (vettore di output). Il vettore `y_pred`, delle previsioni sul periodo di test, viene infine ottenuto tramite il metodo `predict(X_test)` e successivamente visualizzato come dataframe.

4.4.4 Valutazione del modello e generazione dei grafici

Una volta ottenute le previsioni del modello sul periodo di test, è necessario valutarne l'accuratezza e rappresentarne graficamente il comportamento. Le attività principali svolte in quest'ultima parte del progetto sono le seguenti:

- calcolo delle metriche di errore, MAE e RMSE, sul test set;
- confronto del MAE del modello con quello di una *baseline naive*;
- analisi dell'importanza delle feature;
- generazione dei grafici previsionali nel periodo di test, sull'intera serie e sulle previsioni future.

Per iniziare, vengono calcolate due metriche di errore sul periodo di test, il MAE e l'RMSE. Il codice riportato in Figura 4.15 applica le funzioni `mean_absolute_error` e `mean_squared_error` di scikit-learn ai valori di test forniti al modello (`y_test`) e a quelli previsti (`y_pred`).

```
# 7) Valutazione del modello (metriche MAE / RMSE)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("MAE:", mae)
print("RMSE:", rmse)

MAE: 125.56631746031746
RMSE: 170.16897522585597

# 7.1) Calcolo Baseline "Naive"

y_pred_naive = y_test.shift(1) # previsione "naive": valore di questa settimana = valore della settimana precedente

y_pred_naive_aligned = y_pred_naive.iloc[1:] # toglie il primo valore poichè è un NaN prodotto dallo shift(1)
y_test_aligned = y_test.iloc[1:] # toglie il primo valore per avere una serie della stessa lunghezza di y_pred_naive_aligned

mae_naive = mean_absolute_error(y_test_aligned, y_pred_naive_aligned) # MAE del modello "naive"

print(f"MAE Modello (Random Forest): {mae:.2f}")
print(f"MAE Baseline (Naive): {mae_naive:.2f}")

# Confronto
if mae < mae_naive:
    print("\nRISULTATO: Il modello Random Forest è migliore del baseline naive.")
    print(f"E' più accurato di circa {mae_naive - mae:.2f} incidenti a settimana.")
else:
    print("\nATTENZIONE: Il modello Random Forest è peggiore (o uguale) del baseline naive.")

MAE Modello (Random Forest): 125.57
MAE Baseline (Naive): 152.87

RISULTATO: Il modello Random Forest è migliore del baseline naive.
È più accurato di circa 27.30 incidenti a settimana.
```

Figura 4.15: Confronto del MAE del modello appena addestrato con quello di un modello base *naive*, che per ogni settimana prevede lo stesso numero di incidenti della settimana precedente

Si ottengono così:

- $MAE \approx 126$: indica che il numero previsto di incidenti settimanali si discosta in valore assoluto dal numero reale di circa 126 incidenti;
- $RMSE \approx 170$: è la metrica più sensibile agli errori grandi, che, in questo caso, indica che gli scostamenti più rilevanti tra i valori previsti e quelli effettivi si collocano nell'ordine di 170 incidenti settimanali.

Per valutare se le prestazioni del modello siano effettivamente significative, è stato introdotto un modello di confronto semplice, detto *baseline naive*, in cui la previsione per una settimana coincide con il numero di incidenti osservato nella settimana precedente. Il relativo errore viene calcolato applicando nuovamente il MAE dopo aver riallineato le due serie, in modo che abbiano la stessa lunghezza e siano quindi confrontabili. Il MAE della baseline naive risulta pari a circa 153 incidenti a settimana, quindi il Random Forest è più accurato sulle previsioni di circa 27 incidenti settimanali. Tale risultato conferma che il modello predittivo sviluppato in questo lavoro è effettivamente più utile di una strategia elementare che riporta semplicemente il valore della settimana precedente.

Successivamente è stata analizzata l'importanza delle diverse feature utilizzate nel modello, con conseguente visualizzazione grafica tramite la libreria matplotlib (Figura 4.16).

```
# 7.2) Analisi Importanza Feature

# Estrae i contributi delle Features dal modello
importances = rf.feature_importances_
feature_names = X_train.columns # nomi delle features

# Crea una Series pandas per visualizzarle facilmente
feat_imp = pd.Series(importances, index=feature_names).sort_values()

# Crea il grafico
plt.figure(figsize=(10, 6))
feat_imp.plot(kind='barh', title='Importanza delle Feature (Random Forest)')
plt.xlabel('Importanza (basata su Gini)')
plt.ylabel('Feature')
plt.grid(alpha=0.3)
plt.tight_layout() # per non tagliare le etichette
plt.show()
```

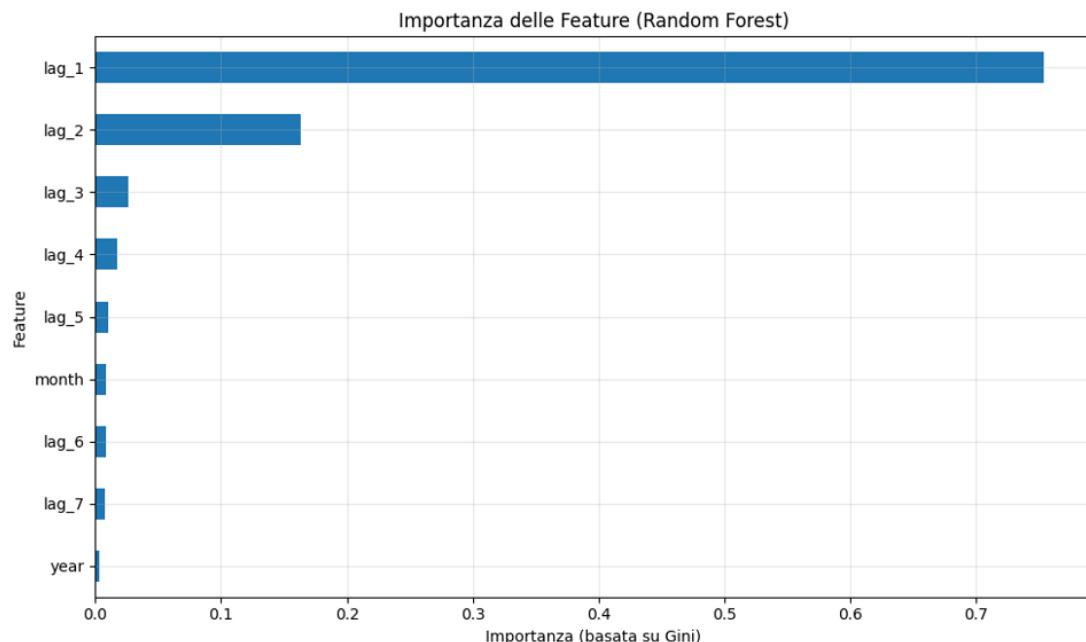


Figura 4.16: Grafico di importanza di tutte le feature utilizzate nel modello

A partire dall'attributo `feature_importances_` del modello, viene costruita una serie Pandas ordinata, che quantifica il contributo relativo di ciascuna variabile alla riduzione dell'impurità nei nodi degli alberi. Il grafico a barre orizzontali associato, evidenzia come le prime due lag (`lag_1` e `lag_2`) siano di gran lunga le più informative; seguono immediatamente le lag successive e la variabile `month`, mentre `year` risulta meno influente. Ciò conferma che, per prevedere il numero di incidenti in una certa settimana, le informazioni più rilevanti sono i valori osservati nelle settimane immediatamente precedenti.

Passando ora alla rappresentazione grafica delle previsioni, viene utilizzato il dataframe `results` che comprende, per ogni settimana del periodo di test, il valore osservato (`y_actual`) e quello previsto dal modello (`y_pred`). La Figura 4.17 mostra il grafico con il solo intervallo di test, composto dalle seguenti due linee:

- *linea blu*: rappresenta il numero di incidenti settimanali osservati tra il 2022 e il 2023;
- *linea arancione*: rappresenta le corrispondenti previsioni, fatte dal Random Forest.

Visivamente, le due serie seguono andamenti simili, con il modello che riesce a catturare le principali oscillazioni della serie, pur commettendo errori più evidenti in corrispondenza dei picchi o delle cadute improvvise.

```
# 8.1) Grafico col solo intervallo di test
plt.figure(figsize=(12, 5))
plt.plot(results.index, results['y_actual'],
          label='Incidenti osservati (test)', linewidth=2)
plt.plot(results.index, results['y_pred'],
          label='Previsione modello (test)', linewidth=2)

plt.xlabel('Data')
plt.ylabel('Numero di incidenti settimanali')
plt.title('Previsione del numero di incidenti settimanali - Random Forest')
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

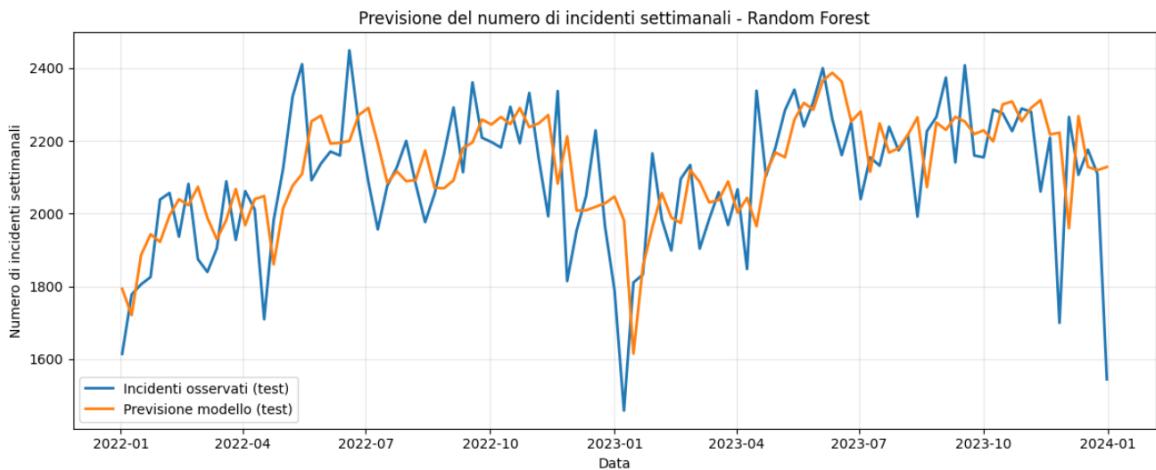


Figura 4.17: Grafico di previsione del numero di incidenti settimanali nel solo intervallo di test

Per dare maggior contesto alle previsioni, la Figura 4.18 estende la visualizzazione all'intera serie storica settimanale compresa tra il 2016 e il 2023. Tale grafico presenta le seguenti linee:

- *linea azzurra trasparente*: rappresenta la serie completa degli incidenti settimanali;
- *linee arancione e verde*: come nel grafico precedente, rappresentano rispettivamente i valori osservati e previsti nel test.

La linea tratteggiata verticale indica l’istante di separazione tra periodo di addestramento e periodo di test (1 gennaio 2022). In questo modo, è possibile confrontare le previsioni non solo con i dati effettivi, ma anche con l’evoluzione globale del fenomeno lungo tutta la finestra temporale considerata.

```
# 8.2) Grafico con il contesto completo Train + Test

plt.figure(figsize=(12, 6))

plt.plot(ts.index, ts['num_crashes'],
         label='Serie storica completa', alpha=0.3)
plt.plot(results.index, results['y_actual'],
         label='Incidenti osservati (test)', linewidth=2)
plt.plot(results.index, results['y_pred'],
         label='Previsione modello', linewidth=2)

split_dt = pd.to_datetime(split_date)
plt.axvline(split_dt, linestyle='--', color='k', label='Inizio periodo di test')

plt.xlabel('Data')
plt.ylabel('Numero di incidenti settimanali')
plt.title('Forecasting del numero di incidenti - modello Random Forest')
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

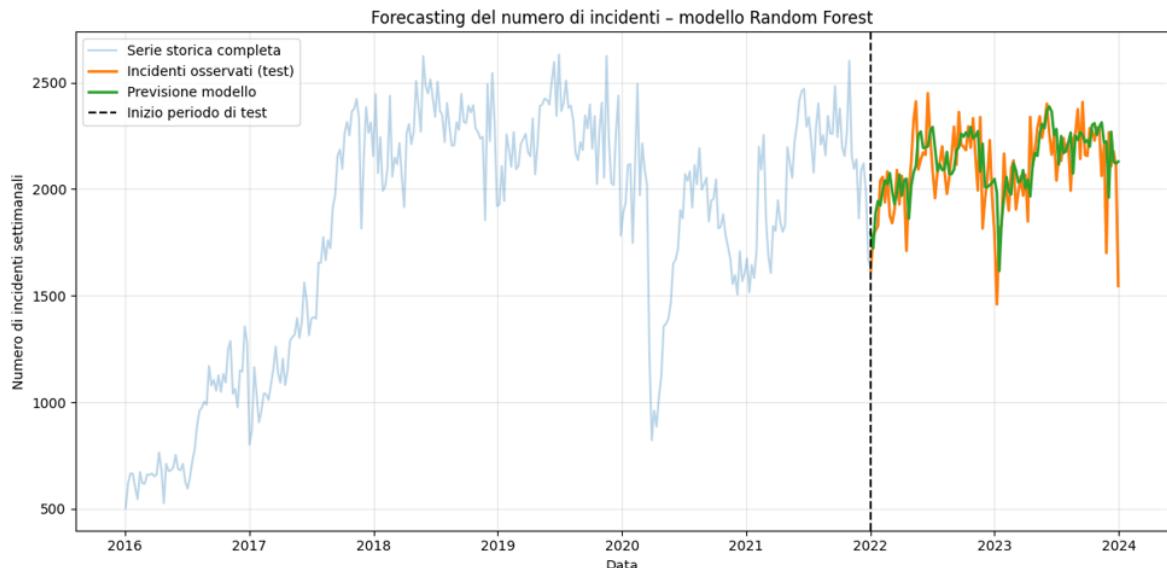


Figura 4.18: Grafico del test esteso all’intero periodo della serie

Dopo aver verificato il comportamento del modello sul periodo di test, si passa all’addestramento finale su tutti i dati disponibili tra il 2016 e il 2023 (Figura 4.19).

```
# 9) Allenamento del modello finale su tutti i dati (2016 - 2023)

X_final = supervised.drop(columns=['y']) # utilizzo dell'intero dataframe 'supervised'
y_final = supervised['y']

# Crea e allena il modello finale
rf_final = RandomForestRegressor(
    n_estimators=300,
    random_state=42,
    n_jobs=-1
)
rf_final.fit(X_final, y_final)
```

Figura 4.19: Allenamento del modello su tutto il periodo della finestra di lavoro scelta (2016-2023)

Per fare ciò, il modello `rf_final` viene istanziato con gli stessi parametri utilizzati in precedenza e poi addestrato sull'intero dataframe supervisionato (`X_final`, `y_final`). In questo modo, l'algoritmo utilizza tutte le informazioni storiche presenti nella finestra di lavoro prima di passare alla fase di previsione.

Per generare le previsioni future, è stata implementata una procedura di *previsione autoregressiva*, illustrata dal codice in Figura 4.20.

```
# 10) Generazione della previsione autoregressiva

N_LAGS = 7           # stesso n_lags usato prima
N_FORECAST = 52      # numero di settimane da prevedere (circa 1 anno)

# Prende gli ultimi lag reali per iniziare la previsione
history_lags = y_final.iloc[-N_LAGS:].tolist()

# Crea l'indice di date per il futuro
last_real_date = y_final.index[-1]
future_dates = pd.date_range(
    start=last_real_date + pd.Timedelta(weeks=1),
    periods=N_FORECAST,
    freq='W'
)

# Lista per salvare le previsioni
future_forecasts = []

print(f"Inizio previsione autoregressiva per {N_FORECAST} settimane...")

for current_date in future_dates:
    # 1- Costruisce le features per la data corrente
    features = {}
    for i in range(1, N_LAGS + 1):
        features[f'lag_{i}'] = history_lags[i]
    features['month'] = current_date.month
    features['year'] = current_date.year

    # 2- Crea un dataframe per la previsione
    X_new = pd.DataFrame(features, index=[current_date])
    X_new = X_new[X_final.columns]

    # 3- Previsione
    new_pred = rf_final.predict(X_new)[0] # [0] per estrarre il singolo valore

    # 4- Salva la previsione e aggiorna la history
    future_forecasts.append(new_pred)
    history_lags.append(new_pred) # La previsione diventa 'storia'
    history_lags = history_lags[-N_LAGS:] # Mantiene solo gli ultimi 7 valori

print("Previsione completata.")

Inizio previsione autoregressiva per 52 settimane...
Previsione completata.
```

Figura 4.20: Codice per la previsione autoregressiva del modello

La logica dietro questo processo può essere riassunta nei seguenti passi:

1. impostazione del numero di lag da utilizzare (`N_LAGS = 7`) e del numero di settimane da prevedere (`N_FORECAST = 52`, circa un anno);
2. estrazione degli ultimi sette valori osservati della serie (`history_lags`), che costituiscono il punto di partenza della previsione;
3. costruzione, per ciascuna settimana futura da prevedere, di un vettore di feature contenente sia le sette lag, ottenute dai valori presenti in `history_lags`, sia le due variabili temporali associate alla settimana corrente;

4. creazione di un nuovo dataframe (`X_new`) con le feature organizzate nella stessa struttura utilizzata in fase di addestramento (`X_final`) e passaggio di tale vettore al modello `rf_final` per ottenere una nuova previsione;
5. salvataggio della previsione ottenuta e aggiornamento della storia (`history_lags`), sostituendo il valore più vecchio con quello appena previsto, così da poter generare la previsione per la settimana successiva.

Tale ciclo si ripete fino a ottenere le 52 previsioni desiderate.

Come parte finale di questo lavoro, la Figura 4.21 mostra il grafico conclusivo dell’analisi predittiva, composto dalle seguenti linee:

- *linea blu*: rappresenta i dati storici osservati tra il 2016 e la fine del 2023;
- *linea rossa*: rappresenta le previsioni autoregressive per le 52 settimane successive, corrispondenti all’anno 2024 e all’inizio del 2025.

```
# 11) Grafico della previsione futura (numero di incidenti nel 2024)

# Trasforma le previsioni in una Series Pandas per costruire più facilmente il grafico
s_forecast = pd.Series(future_forecasts, index=future_dates)

plt.figure(figsize=(14, 7))

plt.plot(y_final, label='Dati storici osservati', color='c0', alpha=0.8) # Dati storici
plt.plot(s_forecast, label='Previsione futura (2024-2025)', color='c3', linestyle='--') # Previsione futura
plt.axvline(last_real_date, linestyle=':', color='k', label='Inizio previsione')

plt.title('Previsione Autoregressiva Incidenti Settimanali')
plt.ylabel('Numero di incidenti settimanali')
plt.xlabel('Data')
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

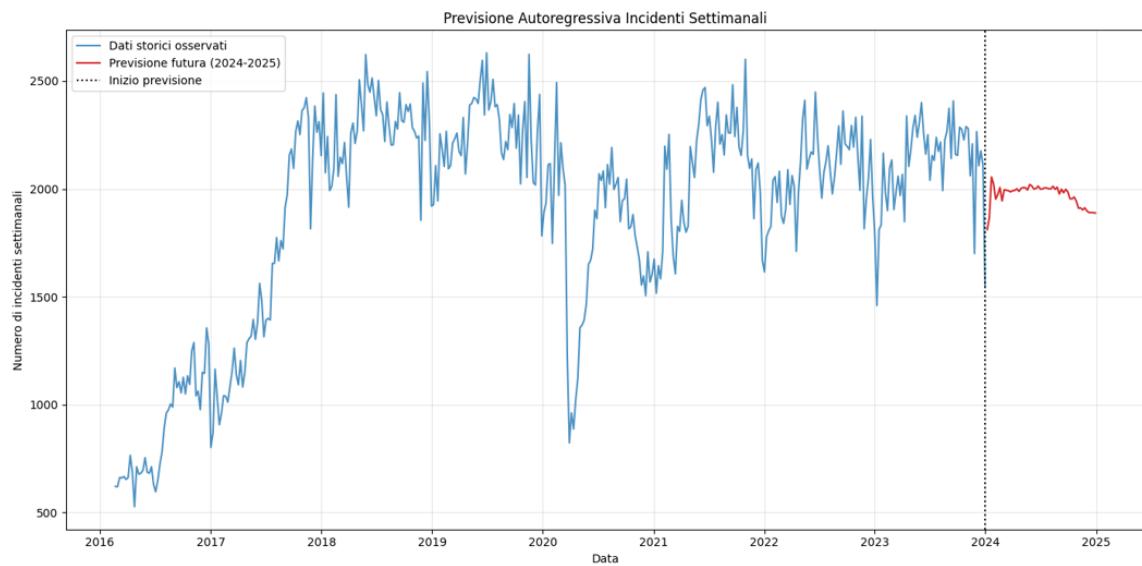


Figura 4.21: Grafico finale dell’analisi predittiva per il 2024 e l’inizio del 2025

Questo grafico rappresenta la conclusione della componente avanzata del progetto e riassume visivamente l’andamento previsto del numero di incidenti settimanali nel periodo successivo ai dati disponibili. Da notare come la curva delle previsioni risulti più regolare e smussata rispetto alla serie storica. Questo accade sia perché il Random Forest, per costruzione, restituisce una media delle previsioni dei singoli alberi decisionali, sia perché la procedura è autoregressiva; ciò significa che ogni valore futuro viene stimato utilizzando anche le previsioni ottenute nei passi precedenti. Di conseguenza, in mancanza di nuove

informazioni esterne, il modello tende ad attenuare i picchi e a riportarsi gradualmente su degli andamenti vicini a quelli osservati nell'ultima parte dei dati storici.

CAPITOLO 5

Discussione in merito al lavoro svolto

In questo capitolo vengono riassunte e discusse tutte le attività svolte nel progetto, evidenziando sia i punti di forza sia le principali criticità emerse. Dopo una breve sintesi del percorso complessivo, l'attenzione si concentra sulle scelte effettuate nella preparazione dei dati, nella progettazione dell'applicativo APEX e nella realizzazione della parte predittiva, citando gli aspetti che hanno maggiormente contribuito alla qualità del lavoro e quelli che potrebbero essere migliorati in sviluppi futuri.

5.1 Valutazione complessiva, punti di forza e criticità

Il progetto documentato in questa tesi è stato interamente incentrato sulla tematica degli incidenti stradali, un ambito ritenuto particolarmente adatto e significativo da affrontare tramite strumenti di gestione, visualizzazione e analisi di dati. Fin dall'inizio, l'obiettivo principale è stato quello di lavorare con un dataset reale di circa 700.000 record riguardanti gli incidenti stradali nella città di Chicago. Il lavoro è iniziato con la progettazione e l'implementazione di un applicativo APEX per gestire e visualizzare i dati in maniera interattiva, fino a terminare con una parte di forecasting, per includere un'analisi più avanzata.

Nel complesso, l'intero flusso di lavoro si è basato su una solida struttura metodologica e tutti gli obiettivi iniziali possono essere considerati raggiunti. L'applicativo APEX permette di gestire, consultare e filtrare i dati tramite griglie interattive, report e dashboard organizzate per tematica. La sezione di analisi avanzata offre un esempio completo di forecasting che parte dalla preparazione dei dati e arriva fino alla valutazione numerica e alla rappresentazione grafica delle previsioni. Per il resto, alcuni aspetti del progetto hanno richiesto una rimodulazione rispetto all'idea originaria, con particolare riferimento all'ambiente utilizzato per l'analisi predittiva, ma ciò non ha compromesso in nessun modo la coerenza del lavoro svolto.

Per quanto riguarda i punti di forza del progetto, si possono evidenziare i seguenti aspetti principali:

- *Scelta e preparazione accurata dei dati:* durante le prima fasi di analisi si è prestata particolare attenzione alla scelta e alla valutazione del dataset, analizzandone la struttura, il grado di pulizia e l'affidabilità delle informazioni. La successiva costruzione di un modello dati relazionale normalizzato e di viste dedicate alla pulizia e al raggruppamento, ha garantito una base solida su cui fondare l'applicativo.

- *Progettazione strutturata dell'applicativo APEX:* l'applicativo è stato progettato in modo da essere facilmente navigabile e intuitivo da utilizzare. Per fare ciò, è stato fatto uno studio approfondito dell'intero ambiente APEX, della sua architettura e degli elementi costitutivi delle sue applicazioni. Sono state definite una home page, un menu di navigazione laterale e delle sezioni distinte per la griglia completa del dataset, per le tabelle di lookup e per le dashboard tematiche. Sono stati inoltre introdotti componenti di filtraggio e liste di valori per arricchire l'interazione con i grafici, selezionando con cura le tipologie di visualizzazioni più adatte a ciascuna domanda analitica.
- *Integrazione tra tecnologie diverse:* il lavoro combina l'ambiente Oracle APEX, per la parte gestionale e di visualizzazione, con un ambiente Python locale basato su JupyterLab, per l'analisi predittiva. Quest'ultimo ha messo insieme strumenti diversi, come la libreria DuckDB per il database e la libreria scikit-learn per i modelli di machine learning, ed è risultato molto leggero e veloce nell'esecuzione delle operazioni. Questo approccio complessivo dimostra, quindi, come un applicativo gestionale possa essere esteso con componenti avanzate di analisi, senza modificare in modo invasivo l'architettura complessiva. Inoltre, è rilevante menzionare come strumenti locali e poco pesanti possano risultare sorprendentemente efficienti nel trattamento di grandi volumi di dati.
- *Approccio metodologico all'analisi predittiva:* l'implementazione della parte del progetto sull'analisi avanzata è stata preceduta da uno studio teorico sui concetti di machine learning, regressione, serie storiche e metriche di valutazione. Il processo di analisi è stato impostato in modo ordinato, passando dalla costruzione e suddivisione della serie storica settimanale, alla definizione degli input per l'addestramento e il test del modello Random Forest. Inoltre, l'introduzione di una baseline naïve ha fornito un confronto chiaro per quantificare il valore aggiunto del modello rispetto a una strategia elementare. La realizzazione dei diversi grafici durante tale processo ha permesso di facilitare la lettura e l'interpretazione dei risultati dell'analisi, rendendoli più comprensibili anche per gli utenti meno esperti sull'argomento.

Dopo quanto appena detto, è ora opportuno evidenziare anche alcune criticità e limiti emersi nel corso del lavoro:

- *Normalizzazione parziale del dataset:* data l'ampia struttura del dataset originale, composto da circa 50 colonne eterogenee, non è stata effettuata una normalizzazione completa del modello relazionale. Si è scelto, infatti, di mantenere alcune parti in forma de-normalizzata, per evitare una frammentazione eccessiva delle tabelle. In particolare, alcuni gruppi di attributi concettualmente correlati, come concuse e ferite, sono stati lasciati come colonne distinte nella stessa tabella, in modo da non complicare troppo la struttura del database. Anche alcune operazioni di aggregazione e formattazione sono state effettuate direttamente nelle query sorgenti dei grafici APEX, invece che in viste dedicate. Sebbene ciò non comprometta la correttezza dei risultati, in una versione successiva sarebbe preferibile centralizzare tali trasformazioni in viste specifiche, migliorando la manutenibilità, la chiarezza e la riusabilità del codice.
- *Vincoli dell'ambiente Oracle:* l'utilizzo di un workspace gratuito ha posto limiti significativi in termini di risorse, impedendo di completare l'analisi predittiva direttamente nell'ambiente dedicato OML4Py. Ciò ha reso necessario migrare l'intera parte di forecasting in un ambiente Python locale. Sebbene questa soluzione abbia portato a risultati soddisfacenti dal punto di vista tecnico, ha ridotto la possibilità di integrare completamente la componente predittiva all'interno dell'applicativo APEX.

- *Limitazioni del dataset e complessità del problema:* il modello di regressione è stato costruito principalmente a partire dal numero di incidenti settimanali, arricchito da due variabili di calendario. La non inclusione di ulteriori variabili esplicative ha limitato la capacità dell'algoritmo di catturare alcune variazioni improvvise osservate nella serie. Infatti, nonostante il modello risulti più accurato di una baseline naïve di circa 27 incidenti a settimana, i valori del MAE e dell'RMSE confermano la difficoltà nel prevedere con precisione un fenomeno fortemente variabile e influenzato da molti altri fattori. Per questo, la componente predittiva del progetto va considerata come un primo prototipo, utile soprattutto a mostrare la fattibilità dell'approccio più che a fornire previsioni precise e di dettaglio.

Nel complesso, questa discussione evidenzia un percorso ampio e strutturato, che parte dalla preparazione di un dataset reale, comprende la costruzione di un gestionale completo in Oracle APEX e termina con la realizzazione di un modello predittivo adeguatamente motivato e valutato. Pur in presenza di limiti legati agli strumenti e ai dati disponibili, l'esperienza mostra come un approccio metodico e attento ai dettagli permetta di integrare in modo efficace componenti gestionali, analitiche e predittive all'interno di un unico progetto.

CAPITOLO 6

Conclusioni

Questo lavoro ha permesso di attraversare l'intero ciclo di vita di un progetto basato sui dati. Partendo dalla scelta del dataset, si è passati alla sua trasformazione in un modello relazionale coerente, arrivando allo sviluppo di un'applicazione gestionale e di una componente di analisi predittiva. Dal punto di vista applicativo, l'uso di Oracle APEX ha richiesto sia l'apprendimento degli aspetti tecnici della piattaforma sia uno studio sull'esperienza utente. Si è, infatti, prestata particolare attenzione alla struttura della navigazione, all'organizzazione delle pagine e alla scelta dei grafici e delle palette cromatiche. Tutto ciò ha portato alla realizzazione di dashboard capaci di guidare l'utente nella lettura delle informazioni, senza sovraccaricare la visualizzazione. La componente predittiva ha introdotto un'ulteriore dimensione al progetto, mostrando anche i vantaggi di un ambiente Python leggero in locale. La costruzione della serie storica, la scelta delle feature, l'addestramento del modello e il suo confronto con una baseline naïve hanno permesso di ottenere previsioni valide e motivate. Dal punto di vista formativo, la tesi ha rappresentato un'occasione per unire in modo concreto la modellazione dei dati, lo sviluppo di applicazioni web e l'implementazione di diversi metodi di analisi. Tra i possibili sviluppi futuri si possono considerare l'integrazione dei risultati predittivi all'interno dell'applicativo APEX, l'arricchimento del dataset con nuove variabili esplicative e l'esplorazione di modelli di machine learning più avanzati, mantenendo sempre come riferimento l'impostazione metodologica che ha guidato questo progetto.

Bibliografia

- AI SAFETY BOOK PROJECT (2023), «Supervised Learning», <https://www.aisafetybook.com/textbook/artificial-intelligence-and-machine-learning>, sezione del testo online dedicata ai fondamenti del supervised learning.
- AMAZON WEB SERVICES (2023), *Differenza tra ETL ed ELT*, AWS, confronto tecnico tra i paradigmi ETL e ELT nei processi di data integration.
- APACHE SOFTWARE FOUNDATION (2004), *Apache License, Version 2.0*, licenza open-source che regola l'uso e la distribuzione di software e dataset.
- APP IN 5 MINUTI (2023a), «Configurare e personalizzare una lista in Oracle APEX», <https://appin5minuti.it/come-configurare-e-personalizzare-una-lista-in-oracle-apex/>, tutorial sulle liste e i menu di navigazione in Oracle APEX.
- APP IN 5 MINUTI (2023b), «Oracle APEX Interactive Grid», <https://appin5minuti.it/oracle-apex-interactive-grid/>, guida alla creazione e gestione delle griglie interattive in APEX.
- APP IN 5 MINUTI (2023c), «Oracle APEX Interactive Report», <https://appin5minuti.it/oracle-apex-interactive-report/>, guida introduttiva alla creazione e configurazione dei report interattivi in APEX.
- APP IN 5 MINUTI (2023d), «Oracle APEX Navigation Menu», <https://appin5minuti.it/oracle-apex-navigation-menu/>, guida alla configurazione del menù di navigazione in Oracle APEX.
- BETTIO, D. (2023), «Valutazione delle prestazioni di un modello di regressione», <https://www.diariodiunanalista.it/posts/valutazione-delle-prestazioni-di-un-modello-di-regressione/>, articolo in italiano sulle principali metriche di errore per modelli di regressione.
- BHAT, S. (2021), «A Comprehensive Guide to Random Forest Regression», <https://medium.com/@bhatshrinath41/a-comprehensive-guide-to-random-forest-regression-43da559342bf>, guida passo-passo al modello di Random Forest per problemi di regressione.
- BNOVA CONSULTING (2023), «Data Quality: a cosa serve», <https://www.bnova.it/data-governance/data-quality-a-cosa-serve/>, articolo divulgativo sui principi e sulle pratiche di Data Quality.

- BROWNLEE, J. (2017), «What Is Time Series Forecasting?», <https://machinelearnin gmaster.com/time-series-forecasting/>, introduzione al forecasting su serie temporali e ai relativi concetti di base.
- CODESIGNAL (2023), «Creating Lag Features for Time Series Prediction», <https://code signal.com/learn/courses/preparing-financial-data-for-machine-l earning/lessons/creating-lag-features-for-time-series-prediction>, lezione online sul concetto di lag features e sulla loro costruzione in ambito time series.
- CORNELL UNIVERSITY (2022), «Statistical Learning», <https://info5940.infosci.cor nell.edu/notes/machine-learn/statistical-learning/>, appunti del corso che introducono il problema di apprendimento statistico e la formulazione $Y = f(X) + \varepsilon$.
- DAGSHUB (2022), «Baseline Models», <https://dagshub.com/glossary/baseline -models/>, definizione e ruolo dei modelli baseline nel processo di valutazione delle prestazioni.
- DATABRICKS (2024), *ETL vs ELT*, articolo tecnico di Databricks che descrive differenze e casi d'uso tra ETL e ELT.
- DATAVIZBLOG (2023), «Chart Literacy: Origins – Dr. Andrew Abela, Part 1 of a Series», <https://datavizblog.com/2023/03/26/choosing-the-right-chart-when-creating-a-data-visualization-part-1-of-a-series/>, approfondimento su come scegliere il tipo di grafico più adatto per rappresentare un dato.
- GEEKSFORGEEKS (2021), «Components of Time Series Data», <https://www.geeksfor geeks.org/data-science/components-of-time-series-data/>, spiegazione delle principali componenti di una serie temporale: trend, stagionalità, ciclicità e rumore.
- GEEKSFORGEEKS (2022a), «ML – Classification vs Regression», <https://www.geeksfor geeks.org/machine-learning/ml-classification-vs-regression/>, articolo che confronta problemi di classificazione e regressione in machine learning.
- GEEKSFORGEEKS (2022b), «Time Series Forecasting as Supervised Learning», <https://www.geeksforgeeks.org/machine-learning/time-series-forecasting-as-supervised-learning/>, articolo che mostra come trattare il forecasting come un problema di apprendimento supervisionato.
- GEMA BUSINESS SCHOOL (2023), «KPI: significato e caratteristiche», <https://www.gema.it/blog/marketing-comunicazione-e-management/kpi-significato/>, articolo esplicativo sui Key Performance Indicators e sul loro utilizzo per misurare obiettivi.
- GOOGLE CLOUD (2024a), «Cos'è l'intelligenza artificiale?», <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=it>, guida introduttiva di Google Cloud sui concetti base di IA.
- GOOGLE CLOUD (2024b), *Cos'è un database relazionale?*, descrizione dei principi fondamentali dei database relazionali.
- IBM CORPORATION (2023a), «Che cos'è la foresta casuale?», <https://www.ibm.com/it-it/think/topics/random-forest>, articolo divulgativo sulla Random Forest e sui suoi utilizzi in classificazione e regressione.

- IBM CORPORATION (2023b), «Che cos'è un albero decisionale?», <https://www.ibm.com/it-it/think/topics/decision-trees>, introduzione IBM ai modelli ad albero decisionale e alle loro applicazioni.
- IBM CORPORATION (2024), «Cos'è l'intelligenza artificiale?», <https://www.ibm.com/it-it/think/topics/artificial-intelligence>, panoramica IBM su intelligenza artificiale, machine learning e principali casi d'uso.
- IBM THINK BLOG (2024), «Cos'è il processo decisionale basato sui dati», <https://www.ibm.com/it-it/think/topics/data-driven-decision-making>, articolo divulgativo sul Data Driven Decision Making e sui suoi benefici per le aziende.
- JOHNY, A. (2023), «Chicago Traffic Crashes Dataset», <https://www.kaggle.com/datasets/anoopjohny/traffic-crashes-crashes/data>, dataset pubblico contenente i dati sugli incidenti stradali della città di Chicago.
- K., N. (2019), «Machine Learning Paradigms for Beginners», <https://medium.com/@nimk/machine-learning-paradigms-for-beginners-9f4c93ae1d5b>, articolo divulgativo che illustra i tre paradigmi principali del machine learning (supervised, unsupervised, reinforcement).
- LEOGRANDE, D. (2017), *L'analisi delle serie storiche per la programmazione delle attività*, Università degli Studi di Bari, appunti didattici sull'analisi delle serie storiche e applicazioni pratiche.
- MICROSOFT CORPORATION (2022), *Principi della normalizzazione dei database*, guida ufficiale Microsoft sulla normalizzazione e le forme normali.
- MLU-EXPLAIN (2020), «Train, Test, and Validation Sets», <https://mlu-explain.github.io/train-test-validation/>, guida didattica alla suddivisione dei dati in train, validation e test set.
- OMARZAI, F. (2022), «Random Forest In-Depth», <https://medium.com/@fraidoonomaarzai99/random-forest-in-depth-f0556817c40b>, articolo di approfondimento sul funzionamento interno delle Random Forest, con schemi esplicativi.
- ORACLE APEX PRODUCT MANAGEMENT TEAM (2024), *Universal Theme – Getting Started*, Oracle Corporation, documentazione ufficiale Oracle APEX sull'uso e la personalizzazione dell'Universal Theme.
- ORACLE CORPORATION (2020), *Developing the Dashboard Page*, Oracle Documentation.
- ORACLE CORPORATION (2022), *Managing Schemas*, Oracle Documentation.
- ORACLE CORPORATION (2023), *Creare un'applicazione APEX da un foglio di calcolo*, guida Oracle per importare e trasformare dati da fogli di calcolo in applicazioni APEX.
- ORACLE CORPORATION (2024a), *Oracle APEX Foundation Course*, Oracle University.
- ORACLE CORPORATION (2024b), *Oracle APEX Foundations Guide*, Oracle University.
- RESEARCHGATE (2020), «Time Series Components», https://www.researchgate.net/figure/Time-Series-components-1-A-time-series-is-a-sequence-of-observations-measured-at_fig1_344658764, schema grafico delle componenti principali di una serie storica.
- RIVERY.IO (2023), «ETL vs ELT Diagrammi e Schemi», <https://rivery.io/blog/etl-vs-elt/>, confronto visivo tra i flussi ETL e ELT.

- SHAH, S. M. (2019), «Why Logistic Regression? Why Not Logistic Classification?», <https://discover.hubpages.com/technology/Why-Logistic-Regression-Why-not-Logistic-Classification>, articolo con rappresentazioni grafiche delle differenze tra regressione e classificazione.
- SMITH, J. (2022), «Comparing ORDS and REST Enabled SQL to GraphQL», *JMJ Cloud Blog*.
- TICOPROF (2021a), «Chiavi candidate e chiavi primarie», <https://ticoprof.wordpress.com/database/progettazione-concettuale-database/chiavi-chiavi-candidate-e-chiavi-primarie/>, spiegazione didattica sul concetto di chiavi in un database relazionale.
- TICOPROF (2021b), «Vincoli di integrità referenziale», <https://ticoprof.wordpress.com/?s=chiave+esterna>, approfondimento sui vincoli di chiave esterna nei database relazionali.
- VISUALITICS (2023), «Principi di Dashboard Design: come creare dashboard efficaci», <https://visualitics.it/principi-di-dashboard-design-come-creare-dashboard-efficaci/>, guida sui principi fondamentali per la progettazione di dashboard chiare e leggibili.
- WIKIPEDIA (2024a), «Albero di decisione», https://it.wikipedia.org/wiki/Albero_di_decisione, voce encicopedica sugli alberi decisionali.
- WIKIPEDIA (2024b), «Intelligenza artificiale», https://it.wikipedia.org/wiki/Intelligenza_artificiale, voce encicpedica introduttiva sull'intelligenza artificiale e le sue principali aree applicative.

Sitografia

- Oracle APEX – sito ufficiale <https://apex.oracle.com/en/>
- Oracle su Wikipedia – https://it.wikipedia.org/wiki/Oracle_Database
- Oracle, “What is a Database” – <https://www.oracle.com/it/database/what-is-database/>
- IBM Think Blog, “Oracle Database: prodotti e servizi” – <https://www.ibm.com/it-it/think/topics/oracle>
- Oracle, “Oracle REST Data Services (ORDS)” – <https://www.oracle.com/it/database/technologies/appdev/rest.html>
- Oracle, “Oracle Cloud Free Tier” – <https://www.oracle.com/it/cloud/free/>
- Oracle, “Machine Learning for the Database” – <https://www.oracle.com/it/artificial-intelligence/database-machine-learning/>
- Oracle APEX Learn, “SQL Workshop” – <https://apex.oracle.com/en/learn/getting-started/sql-workshop/>
- Kaggle, “Chicago Traffic Crashes Dataset” – <https://www.kaggle.com/datasets/anoopjohny/traffic-crashes-crashes/data>
- Apache Software Foundation, “Apache License 2.0” – <https://www.apache.org/licenses/LICENSE-2.0>
- AWS, “Differenza tra ETL ed ELT” – <https://aws.amazon.com/it/compare/the-difference-between-etl-and-elt/>
- Databricks, “ETL vs ELT” – <https://www.databricks.com/it/discover/etl-vs-elt>
- Rivery.io, “ETL vs ELT: diagrammi e schemi” – <https://rivery.io/blog/etl-vs-elt/>
- Oracle, “Creare un’applicazione APEX da un foglio di calcolo” – https://docs.oracle.com/it/learn/apex_createapp_spreadsheet/
- Google Cloud, “Cos’è un database relazionale?” – <https://cloud.google.com/learn/what-is-a-relational-database?hl=it>

- Microsoft, "Principi della normalizzazione dei database" – <https://learn.microsoft.com/it-it/office/troubleshoot/access/database-normalization-description>
- Ticoprof, "Chiavi candidate e chiavi primarie" – <https://ticoprof.wordpress.com/database/progettazione-concettuale-database/chiavi-chiavi-candidate-e-chiavi-primarie/>
- Ticoprof, "Vincoli di integrità referenziale" – <https://ticoprof.wordpress.com/?s=chiave+esterna>
- Bnova, "Data Quality: a cosa serve" – <https://www.bnova.it/data-governance/data-quality-a-cosa-serve/>
- App in 5 Minuti, "Oracle APEX Interactive Report" – <https://appin5minuti.it/oracle-apex-interactive-report/>
- App in 5 Minuti, "Oracle APEX Interactive Grid" – <https://appin5minuti.it/oracle-apex-interactive-grid/>
- App in 5 Minuti, "Liste in Oracle APEX" – <https://appin5minuti.it/come-configure-e-personalizzare-una-lista-in-oracle-apex/>
- App in 5 Minuti, "Oracle APEX Navigation Menu" – <https://appin5minuti.it/oracle-apex-navigation-menu/>
- IBM Think Blog, "Cos'è il processo decisionale basato sui dati" – <https://www.ibm.com/it-it/think/topics/data-driven-decision-making>
- Visualitics, "Principi di Dashboard Design: come creare dashboard efficaci" – <https://visualitics.it/principi-di-dashboard-design-come-creare-dashboards-efficaci/>
- DatavizBlog, "Chart Literacy: Origins – Dr. Andrew Abela, Part 1 of a Series" – <https://datavizblog.com/2023/03/26/choosing-the-right-chart-when-creating-a-data-visualization-part-1-of-a-series/>
- GEMA Business School, "KPI: significato e caratteristiche" – <https://www.gema.it/blog/marketing-comunicazione-e-management/kpi-significato/>
- Oracle APEX Team, "Universal Theme – Getting Started" – https://oracleapex.com/ords/r/apex_pm/ut/getting-started
- Wikipedia, "Intelligenza artificiale" – https://it.wikipedia.org/wiki/Intelligenza_artificiale
- IBM Think Blog, "Artificial Intelligence" – <https://www.ibm.com/it-it/think/topics/artificial-intelligence>
- Google Cloud, "Cos'è l'intelligenza artificiale?" – <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=it>
- Medium, "Machine Learning Paradigms for Beginners" – <https://medium.com/@nimk/machine-learning-paradigms-for-beginners-9f4c93ae1d5b>
- AI Safety Book, "Supervised Learning" – <https://www.aisafetybook.com/textbook/artificial-intelligence-and-machine-learning>

- Cornell University, "Statistical Learning" – <https://info5940.infosci.cornell.edu/notes/machine-learn/statistical-learning/>
- GeeksforGeeks, "ML – Classification vs Regression" – <https://www.geeksforgeeks.org/machine-learning/ml-classification-vs-regression/>
- HubPages, "Why Logistic Regression? Why Not Logistic Classification?" – <https://discover.hubpages.com/technology/Why-Logistic-Regression-Why-not-Logistic-Classification>
- Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning" – <https://esl.hohoweiya.xyz/book/The%20Elements%20of%20Statistical%20Learning.pdf>
- GeeksforGeeks, "Components of Time Series Data" – <https://www.geeksforgeeks.org/data-science/components-of-time-series-data/>
- ResearchGate, "Time Series Components" – https://www.researchgate.net/figure/Time-Series-components-1-A-time-series-is-a-sequence-of-observations-measured-at_fig1_344658764
- Leogrande D., "L'analisi delle serie storiche per la programmazione delle attività" – <https://www.uniba.it/it/docenti/leogrande-domenico/attività-didattica/7Lanalisi delle serie storiche.pdf>
- Machine Learning Mastery, "What Is Time Series Forecasting?" – <https://machingelearningmastery.com/time-series-forecasting/>
- GeeksforGeeks, "Time Series Forecasting as Supervised Learning" – <https://www.geeksforgeeks.org/machine-learning/time-series-forecasting-as-supervised-learning/>
- MLU-Explain, "Train, Test, and Validation Sets" – <https://mlu-explain.github.io/train-test-validation/>
- Diario di un Analista, "Valutazione delle prestazioni di un modello di regressione" – <https://www.diariodianalista.it/posts/valutazione-delle-prestazioni-di-un-modello-di-regressione/>
- DagsHub, "Baseline Models" – <https://dagshub.com/glossary/baseline-models/>
- CodeSignal, "Creating Lag Features for Time Series Prediction" – <https://codesignal.com/learn/courses/preparing-financial-data-for-machine-learning/lessons/creating-lag-features-for-time-series-prediction>
- Wikipedia, "Albero di decisione" – https://it.wikipedia.org/wiki/Albero_di_decisione
- IBM Think Blog, "Che cos'è un albero decisionale?" – <https://www.ibm.com/it-it/think/topics/decision-trees>
- IBM Think Blog, "Che cos'è la foresta casuale?" – <https://www.ibm.com/it-it/think/topics/random-forest>

- Medium, “A Comprehensive Guide to Random Forest Regression” – <https://medium.com/@bhatshrinath41/a-comprehensive-guide-to-random-forest-regression-43da559342bf>
- Medium, “Random Forest In-Depth” – <https://medium.com/@fraidoonmarza199/random-forest-in-depth-f0556817c40b>