Data Wrangling Avanzato e Integrazione SQL da Python

Benvenuti a questo viaggio nel mondo del data wrangling avanzato, un processo fondamentale che trasforma i dati grezzi in informazioni preziose. Esploreremo come questa disciplina sia essenziale per l'analisi dei dati e per prendere decisioni basate sui dati, rappresentando un passaggio cruciale nei campi della data science, del machine learning e della business intelligence.

by Simone Pipitone

Fasi Chiave del Data Wrangling

Il processo di data wrangling si articola in quattro fasi fondamentali che permettono di trasformare dati caotici in risorse preziose:

Data Acquisition

Raccolta di dati da fonti eterogenee come API, database, file Excel, sorgenti web e sistemi legacy.

Data Cleaning

Identificazione e rimozione di errori, duplicati, valori anomali e inconsistenze nei dati raccolti.

Data Transformation

Standardizzazione del formato e della struttura per preparare i dati all'analisi.

Data Mapping

Consolidamento di diverse fonti in un unico dataset coerente e utilizzabile.

Processo Avanzato: Sei Passaggi Dettagliati

Data Discovery

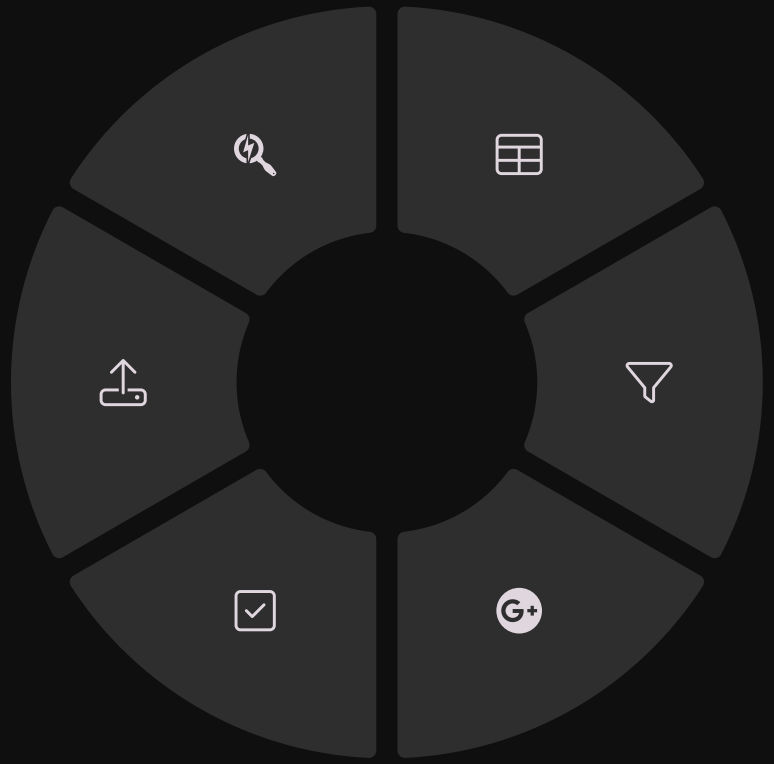
Esplorazione approfondita dei dati grezzi per comprenderne struttura, formato e potenziali problemi.

Data Publishing

Distribuzione dei dati puliti e pronti per l'analisi agli stakeholder.

Data Validation

Verifica della conformità ai requisiti di qualità e coerenza stabiliti.



Data Structuring

Riorganizzazione dei dati in formati utili e manipolabili con trasformazioni strutturali.

Pulizia

Gestione di valori nulli, outlier e normalizzazione dei dati per migliorarne la qualità.

Arricchimento

Integrazione con fonti esterne per aumentare il valore informativo del dataset.

Strumenti e Tecniche Avanzate

Librerie Python Specializzate

- Pandas: manipolazione dataframe, join, pivot, melt
- NumPy: calcolo vettorizzato e manipolazione array
- Beautiful Soup: parsing HTML e scraping web
- PyJanitor: pulizia dati semplificata

Tecniche Avanzate

- Regex per estrazione e pulizia automatica
- Parsing complesso: XML, JSON, CSV non standard
- Profiling dati con pandas-profiling
- Pipeline di trasformazione automatizzate

Case Study: Data Wrangling su Dataset Reale

Analizziamo un caso reale di data wrangling su un dataset di transazioni bancarie con problematiche comuni:

1 Problema: Date in formati inconsistenti

Soluzione: Standardizzazione con **pd.to_datetime()** e gestione di formati multipli tramite parser personalizzati.

2 Problema: Valori monetari mancanti

Soluzione: Utilizzo di **fillna()** con strategie come media mobile, interpolazione o sostituzione condizionale.

3 Problema: Riconciliazione tra sistemi

Soluzione: Implementazione di **pd.merge()** con strategie left, right e outer join per integrare fonti diverse.

Best Practice e Errori Comuni

Best Practice

- Versionamento dei dati con DVC o Git LFS
- Creazione di audit trail per ogni trasformazione
- Validazione progressiva dopo ogni fase
- Documentazione dettagliata delle trasformazioni
- Automazione dei processi ripetitivi

Errori da Evitare

- Introdurre bias durante la pulizia dei dati
- Perdere tracciabilità delle modifiche
- Applicare trasformazioni non reversibili
- Trascurare l'impatto delle modifiche sull'analisi

Data Wrangling e Data Analytics Avanzata

Preprocessing per Machine Learning

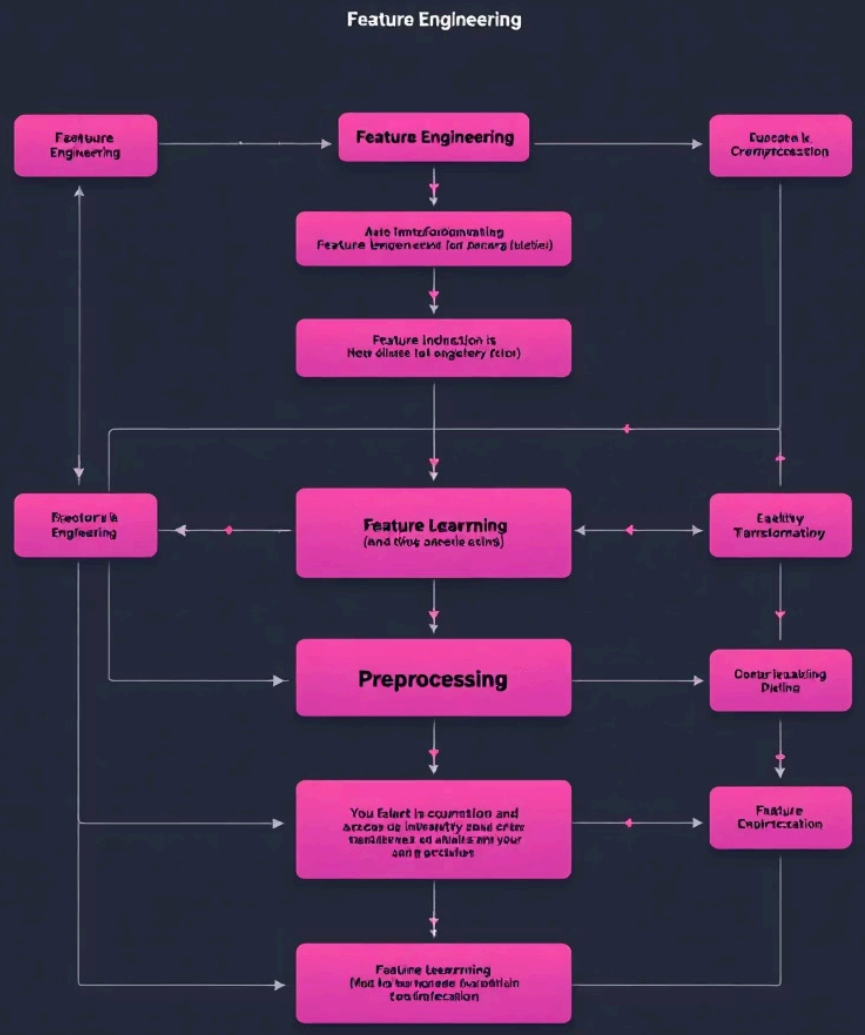
Normalizzazione, codifica categoriale e scaling sono essenziali per preparare i dati per algoritmi ML. L'uso di scikit-learn con pipeline automatizza questi passaggi.

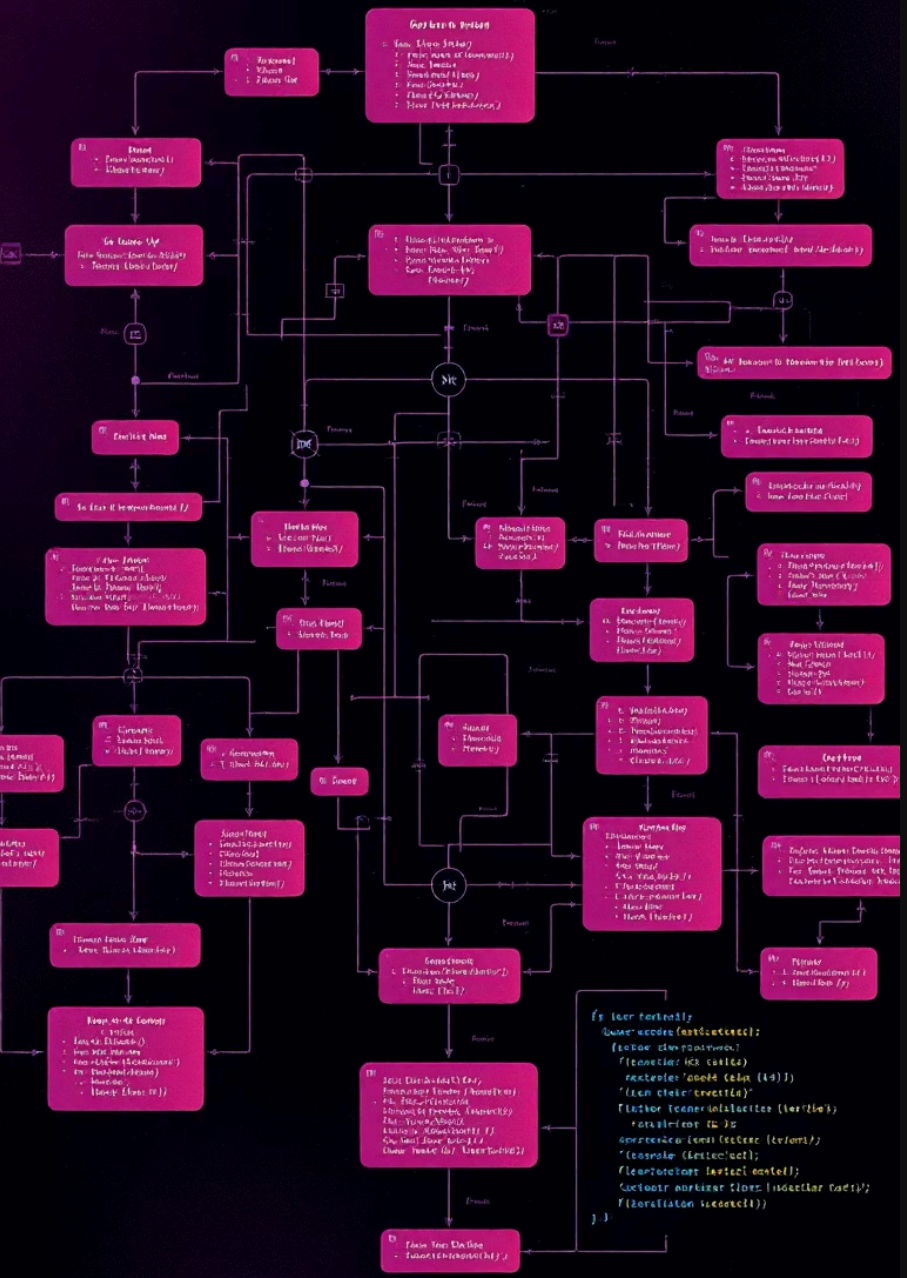
Feature Engineering

Creazione di nuove variabili strategiche che catturano relazioni complesse nei dati. Tecniche come polynomial features, date extraction e text vectorization aumentano il potere predittivo.

Aggregazione per Business Intelligence

Trasformazione dei dati granulari in indicatori di performance chiave attraverso groupby, pivot e aggregazioni temporali per creare dashboard interattive.





Preparare i Dati per l'Integrazione con SQL

Per trasferire efficacemente i dati da Python a un database SQL, è necessario strutturarli secondo schemi coerenti e ben definiti:

1 Normalizzazione delle Colonne

Standardizzazione dei nomi di colonna secondo convenzioni SQL (snake_case), rimozione di caratteri speciali e spazi, definizione di tipi di dato appropriati.

2 Definizione di Chiavi

Identificazione e creazione di chiavi primarie uniche, verifica dell'integrità referenziale per le chiavi esterne e impostazione di vincoli di unicità.

3 Conformità allo Schema

Trasformazione dei tipi di dati in Python per garantire compatibilità con i tipi SQL (datetime, decimal, varchar), gestione di dimensioni massime e valori predefiniti.

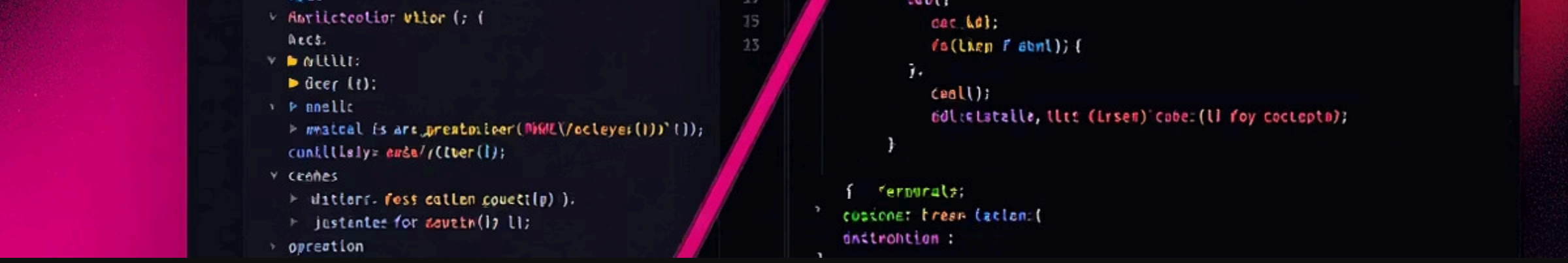
Introduzione all'Interazione tra Python e SQL

Perché Usare SQL da Python?

L'integrazione SQL-Python combina la potenza di elaborazione dei database relazionali con la flessibilità di Python, permettendo di gestire grandi volumi di dati e sfruttare query ottimizzate.

Librerie per Connessione

- **SQLAlchemy**: ORM completo per interazioni di alto livello
- **PyMySQL/psycopg2**: Connettori specifici per database
- **sqlite3**: Integrato in Python per database leggeri
- **pandas.read_sql**: Lettura diretta in DataFrame



Esempio Pratico: Query SQL e Manipolazione da Python

```
import pandas as pd
from sqlalchemy import create_engine

# Creazione connessione al database
engine = create_engine('mysql+pymysql://user:password@localhost/database')

# Creazione tabella da DataFrame
df = pd.DataFrame({
    'id': range(1, 6),
    'nome': ['Anna', 'Marco', 'Lucia', 'Paolo', 'Giulia'],
    'importo': [1200.50, 950.75, 2300.00, 1450.25, 3100.80]
})

# Inserimento dati nel database
df.to_sql('transazioni', engine, if_exists='replace', index=False)

# Lettura dati con query SQL
query = "SELECT nome, importo FROM transazioni WHERE importo > 1500"
risultato = pd.read_sql(query, engine)

print(risultato)
```

Questo esempio mostra come creare una tabella, inserire dati da un DataFrame e quindi eseguire una query SQL per filtrare e recuperare informazioni specifiche. SQLAlchemy gestisce automaticamente la connessione e la traduzione tra i tipi di dati Python e SQL.