# Data Collection & Enrichment Roadmap

This document translates the high‑level suggestions from the **Future Plans** document into a concrete action plan for expanding and enriching the data used in training\_lead\_generation\_model.py. The relevant section of the plan emphasises three activities: expanding data sources, enabling real‑time data ingestion and implementing data‑quality monitoring[[1]](https://github.com/ClaudioLutz/Mitgliedschafts-Voraussage/blob/main/documentation/future_plans.md#L5-L9). The guidance below details how each activity can be implemented in a way that integrates seamlessly with the existing Python training pipeline.

## 1 Expand data sources

The current training pipeline reads monthly snapshots from a Microsoft SQL Server schema via load\_modeling\_data() and load\_current\_snapshot(). To broaden the feature space you should ingest additional data streams:

1. **CRM records (Microsoft Dynamics 365)** – the existing pipeline assumes that membership status and transaction history reside in your SQL tables. To enrich member profiles with sales pipeline data such as lead stage, marketing campaign responses or interaction timestamps, query the Dynamics 365 Web API and join the results by CrefoID. Below is an example using the OData endpoint with OAuth 2.0 authentication. Replace the placeholder tenant/domain information with your own.

import os  
import requests  
import pandas as pd  
from adal import AuthenticationContext  
  
DYNAMICS\_BASE\_URL = "https://YOUR\_ORG.api.crm4.dynamics.com/api/data/v9.2"  
CLIENT\_ID = os.environ['DYNAMICS\_CLIENT\_ID']  
CLIENT\_SECRET = os.environ['DYNAMICS\_CLIENT\_SECRET']  
TENANT\_ID = os.environ['AZURE\_TENANT\_ID']  
  
def get\_dynamics\_token() -> str:  
 """Authenticate against Azure AD and return a bearer token."""  
 auth\_ctx = AuthenticationContext(f"https://login.microsoftonline.com/{TENANT\_ID}")  
 token\_response = auth\_ctx.acquire\_token\_with\_client\_credentials(  
 resource=DYNAMICS\_BASE\_URL,  
 client\_id=CLIENT\_ID,  
 client\_secret=CLIENT\_SECRET  
 )  
 return token\_response['accessToken']  
  
def load\_crm\_data(entity: str = 'leads') -> pd.DataFrame:  
 """Download CRM entity records and return a DataFrame."""  
 token = get\_dynamics\_token()  
 headers = {  
 'Authorization': f'Bearer {token}',  
 'OData-MaxVersion': '4.0',  
 'OData-Version': '4.0',  
 'Accept': 'application/json'  
 }  
 url = f"{DYNAMICS\_BASE\_URL}/{entity}?$select=accountid,fullname,crefoid,new\_stage,new\_annualrevenue"  
 records = []  
 while url:  
 resp = requests.get(url, headers=headers)  
 resp.raise\_for\_status()  
 data = resp.json()  
 records.extend(data['value'])  
 url = data.get('@odata.nextLink') # follow pagination  
 return pd.DataFrame.from\_records(records)  
  
def enrich\_with\_crm(model\_df: pd.DataFrame, crm\_df: pd.DataFrame) -> pd.DataFrame:  
 """Join CRM features (e.g. lead stage, annual revenue) to modelling snapshots."""  
 cols\_to\_use = ['crefoid', 'new\_stage', 'new\_annualrevenue']  
 crm\_df = crm\_df[cols\_to\_use].rename(columns={'crefoid': 'CrefoID'})  
 # Left‑join to preserve all modelling rows; assume CrefoID is unique per company  
 merged = model\_df.merge(crm\_df, on='CrefoID', how='left')  
 return merged

1. **Alternative CRMs** – if Dynamics 365 proves expensive or inflexible, consider alternatives.
2. *HubSpot CRM* offers a free tier with intuitive UI and easy integrations. It provides contact, company and deal management, email tracking and marketing tools[[2]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=HubSpot%20CRM%20is%20one%20of,quickly%20run%20up%20the%20bill). The official [hubspot-api-client](https://github.com/HubSpot/hubspot-api-python) library makes it straightforward to pull contact and deal data via Python.
3. *Pipedrive* focuses on visual pipeline management and sales productivity, with customizable pipelines, automation, email and call tracking[[3]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Pipedrive%20is%20a%20CRM%20designed,get%20straight%20to%20the%20point). The [pipedrive-python](https://github.com/ramon-chivite/pipedrive-python-api) library can be used in a similar fashion.
4. *Odoo* provides a modular open‑source suite covering CRM and ERP processes[[4]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Odoo%3A%20a%20complete%20open%20source,management%20suite). For teams requiring a self‑hosted solution, its odoorpc Python library can authenticate and fetch leads or opportunities.
5. *Nimble CRM* is lightweight and integrates well with Microsoft 365 and Google Workspace, enriching contacts with social‑network data[[5]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Nimble%20CRM%20is%20a%20lightweight,with%20data%20from%20social%20networks).

When opting for a new CRM, ensure the chosen platform exposes a REST API or SDK, then implement a load\_crm\_data() function analogous to the Dynamics example above.

1. **Website analytics and digital interactions** – integrate page‑view metrics, click events or campaign responses from tools like Google Analytics (GA4), Matomo or your email marketing platform. Use their REST APIs or client libraries (e.g., google-analytics-data for GA4) to pull daily or session‑level summaries keyed by user or CrefoID. Join these metrics on the company identifier in the same way as CRM data.
2. **External demographic datasets** – enrich profiles with socio‑economic indicators such as industry benchmarks, company size, or regional statistics. Sources include national statistical offices (e.g., Federal Statistical Office of Switzerland), open‑data portals or commercial providers. Standardize keys (e.g., postal code or NOGA/NACE code) and merge them into the modelling frame before feature engineering.

After loading additional sources, call enrich\_with\_crm() or similar functions prior to the existing feature engineering step:

# Existing pipeline  
engine = make\_engine(SERVER, DATABASE)  
model\_df = load\_modeling\_data(engine, horizon\_months=HORIZON\_MONTHS)  
  
# New: pull CRM data and merge  
crm\_df = load\_crm\_data(entity='leads') # or from HubSpot/Pipedrive  
model\_df = enrich\_with\_crm(model\_df, crm\_df)  
  
# Continue with temporal features, column grouping, etc.  
model\_df = temporal\_feature\_engineer(model\_df)  
num\_cols, low\_card, high\_card = auto\_column\_groups(model\_df)

## 2 Real‑time data ingestion

To rank prospects based on the latest behaviour, ingest events as they occur. Your current code builds monthly snapshots using SQL queries with GETDATE(). For more granular, real‑time features:

1. **Set up a streaming platform** – deploy Apache Kafka, AWS Kinesis or Azure Event Hubs to capture events such as website visits, form submissions or chat interactions. Configure producers within your web application to publish event payloads containing the company identifier and relevant metadata.
2. **Stream processing in Python** – use the confluent-kafka library or kafka-python to consume events and update an in‑memory or persistent store. The following example shows how to consume events and append them to a Parquet file used by the training pipeline:

from confluent\_kafka import Consumer  
import json  
import pandas as pd  
import pyarrow.parquet as pq  
import pyarrow as pa  
  
def consume\_events(topic: str, group\_id: str, bootstrap\_servers: str):  
 consumer\_config = {  
 'bootstrap.servers': bootstrap\_servers,  
 'group.id': group\_id,  
 'auto.offset.reset': 'latest'  
 }  
 consumer = Consumer(consumer\_config)  
 consumer.subscribe([topic])  
 try:  
 while True:  
 msg = consumer.poll(1.0)  
 if msg is None:  
 continue  
 if msg.error():  
 log.warning(f"Consumer error: {msg.error()}")  
 continue  
 event = json.loads(msg.value().decode('utf-8'))  
 # transform to DataFrame; assume event has 'CrefoID' and 'event\_type'  
 df\_event = pd.DataFrame([event])  
 # append to local Parquet file partitioned by date  
 table = pa.Table.from\_pandas(df\_event)  
 pq.write\_to\_dataset(table, root\_path="./stream\_data", partition\_cols=["event\_date"])  
 finally:  
 consumer.close()

1. **Integrate streamed features** – schedule a job that periodically aggregates the streamed event data (e.g., count of visits in the last week) and merges it into the modelling dataset via CrefoID before training. This ensures the model captures up‑to‑date engagement levels.

## 3 Data quality monitoring

Training on unreliable data will harm model performance. Implement checks to flag missing or anomalous values before passing data to the estimator. The following function can be added near the top of training\_lead\_generation\_model.py and invoked after assembling the dataset:

def validate\_data(df: pd.DataFrame) -> None:  
 """Raise warnings or errors for common data‑quality issues."""  
 # 1. Missing value summary  
 missing = df.isna().mean().sort\_values(ascending=False)  
 high\_missing = missing[missing > 0.3] # more than 30 % missing  
 if not high\_missing.empty:  
 log.warning(f"Columns with high missingness (>30%): {high\_missing.index.tolist()}")  
 # 2. Duplicate check  
 duplicates = df.duplicated(subset=['CrefoID', 'snapshot\_date']).sum()  
 if duplicates:  
 log.warning(f"Found {duplicates} duplicate records on CrefoID and snapshot\_date")  
 # 3. Range checks for numeric fields  
 if (df['Umsatz'] < 0).any():  
 raise ValueError("Negative revenue values detected")  
 if (df['MitarbeiterBestand'] < 0).any():  
 raise ValueError("Negative employee counts detected")  
 # Additional domain‑specific checks can be added here

Call validate\_data(model\_df) right after you finish enriching the dataset. If warnings or errors arise, investigate the upstream data pipelines to correct the issues before training.

## 4 Choosing an alternative CRM

When evaluating whether to move away from Microsoft Dynamics 365, consider the needs of your organisation:

* **HubSpot CRM** – widely adopted thanks to its generous free tier, intuitive interface and seamless integrations[[2]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=HubSpot%20CRM%20is%20one%20of,quickly%20run%20up%20the%20bill). Suitable for teams that prioritise accessibility and quick deployment.
* **Pipedrive** – offers a visual and highly intuitive sales pipeline with task automation, email and call tracking and dozens of integrations[[3]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Pipedrive%20is%20a%20CRM%20designed,get%20straight%20to%20the%20point). Ideal for SMEs focused on closing deals efficiently.
* **Odoo** – an open‑source ERP/CRM suite offering full control and modularity[[4]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Odoo%3A%20a%20complete%20open%20source,management%20suite). Best for organisations willing to invest in custom configuration and possibly self‑hosting.
* **Nimble** – integrates tightly with Microsoft 365 and Google Workspace and enriches contacts with social‑network information[[5]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Nimble%20CRM%20is%20a%20lightweight,with%20data%20from%20social%20networks). Good for small teams that rely on email and social interactions.

These alternatives all provide REST APIs or Python SDKs, making it straightforward to implement a load\_crm\_data() function similar to the example above. Evaluate pricing, feature coverage, integration complexity and compliance requirements before migrating.

## Next steps

1. **Prototype CRM integration** – implement and test load\_crm\_data() with your current Dynamics 365 instance or a chosen alternative. Verify that new fields (e.g., lead stage, campaign responses) join correctly on CrefoID.
2. **Set up streaming pipelines** – work with your web‑development team to publish events to Kafka or the chosen streaming platform. Build a consumer that writes events into a format easily ingested by Python (e.g., Parquet).
3. **Develop data‑quality checks** – integrate validate\_data() into your training script and monitor logs for anomalies. Regularly update the checks as new sources are added.
4. **Iterate and evaluate** – once the enriched dataset is incorporated, retrain the lead‑generation model and compare precision and recall metrics. Adjust feature engineering and modelling strategies as needed.

By executing these steps you will enhance the predictive power of the lead‑generation model while ensuring the underlying data remains reliable and reflective of real‑time prospect behaviour.

[[1]](https://github.com/ClaudioLutz/Mitgliedschafts-Voraussage/blob/main/documentation/future_plans.md#L5-L9) GitHub

<https://github.com/ClaudioLutz/Mitgliedschafts-Voraussage/blob/main/documentation/future_plans.md>

[[2]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=HubSpot%20CRM%20is%20one%20of,quickly%20run%20up%20the%20bill) [[3]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Pipedrive%20is%20a%20CRM%20designed,get%20straight%20to%20the%20point) [[4]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Odoo%3A%20a%20complete%20open%20source,management%20suite) [[5]](https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative#:~:text=Nimble%20CRM%20is%20a%20lightweight,with%20data%20from%20social%20networks) 15 alternatives to Microsoft Dynamics 365: the guide to finding the ideal CRM

<https://www.appvizer.com/magazine/customer/client-relationship-mgt/microsoft-dynamics-365-alternative>