CatDB: Data-catalog-guided, LLM-based Generation of Data-centric ML Pipelines

Saeed Fathollahzadeh [1], Essam Mansour [1], and Matthias Boehm [2,3]

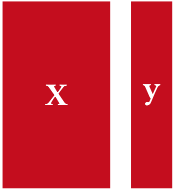
1 Concordia University

2 TU Berlin 3 BIFOLD

September 1-5

VLDB 2025 LONDON

Data-centric ML Pipelines - Background

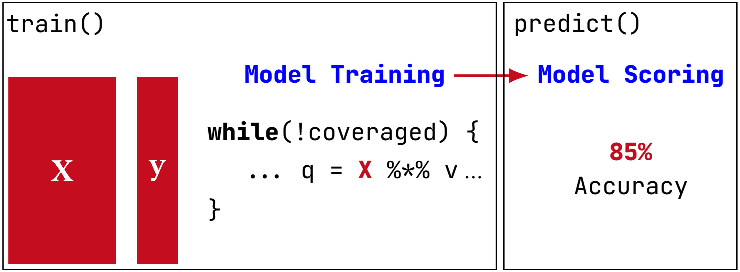


Model Training → **Model Scoring**

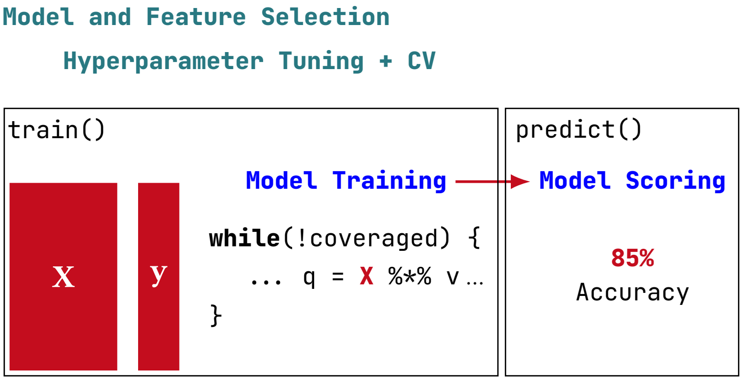
```
while(!coveraged) {
   ... q = X %*% v...
}
```

85%
Accuracy

Data-centric ML Pipelines - Background



Data-centric ML Pipelines - Background



Data-centric ML Pipelines - Background

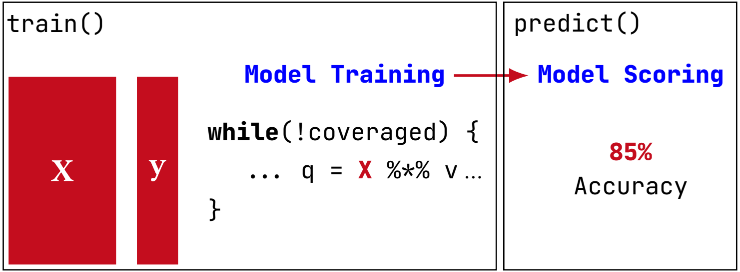
Data Preparation

Data Integration & Data Cleaning

Data Programming & Augmentation

Model and Feature Selection

Hyperparameter Tuning + CV



Data-centric ML Pipelines - Background

Data Preparation

Data Integration & Data Cleaning

Data Programming & Augmentation

Model and Feature Selection

Hyperparameter Tuning + CV


ImageNet Classification with Deep Convolutional Neural Networks

Abstract

1 Introduction

[e.g., Alex Krizhevsky **ImageNet 2012** Challenge **Winner**]

train()

X

y

Model Training → **Model Scoring**

```
while(!coveraged) {
   ... q = X %*% v ...
}
```

predict()

85%
Accuracy

Motivation

Raw Data

Table #1		
c_1	c_2	...
1	a	...
2	-	...
...	...	...

Table #2		
c_1	c_2	...
A	0	..
B	1	...
...	..	...

. . . .

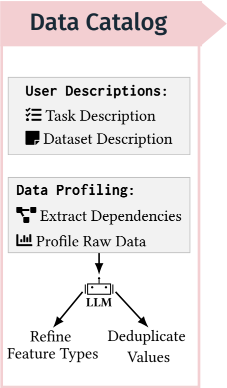
Table #N			
c_1	c_2	...	target (y)
1	A	...	Yes
2	0	...	No
...	...	...	...

How to **Automatically** and **Efficiently** Build a **Data-centric ML Pipeline?**

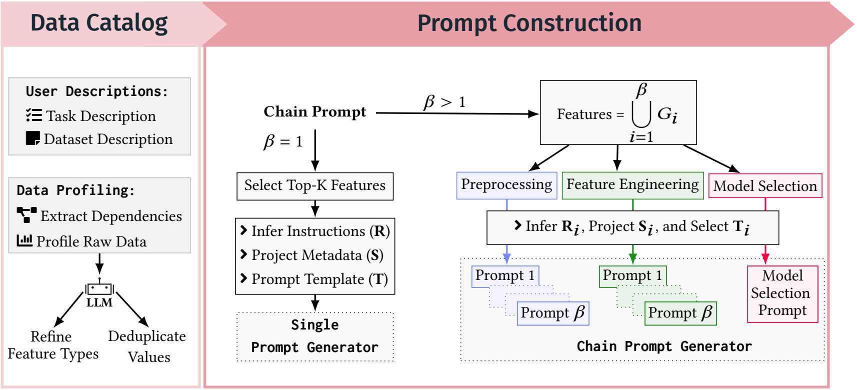
ML Pipeline

```
import pandas as pd
import SimpleImputer
import OneHotEncoder
import ColumnTransformer
import Pipeline
import RandomForestRegressor
import r2_score

trai = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

ca = ["Experience", "Gender"]
cat = Pipeline(steps=[
      ("imputer", SimpleImputer(....
      ("onehot", OneHotEncoder(...

preprocessor = ColumnTransformer(
      transformers = [("cat",...)]
model = RandomForestRegressor(...)
p = Pipeline(steps=[ ....])

p.fit(X_train, y_train)
y_test_pred = p.predict(X_test)
```

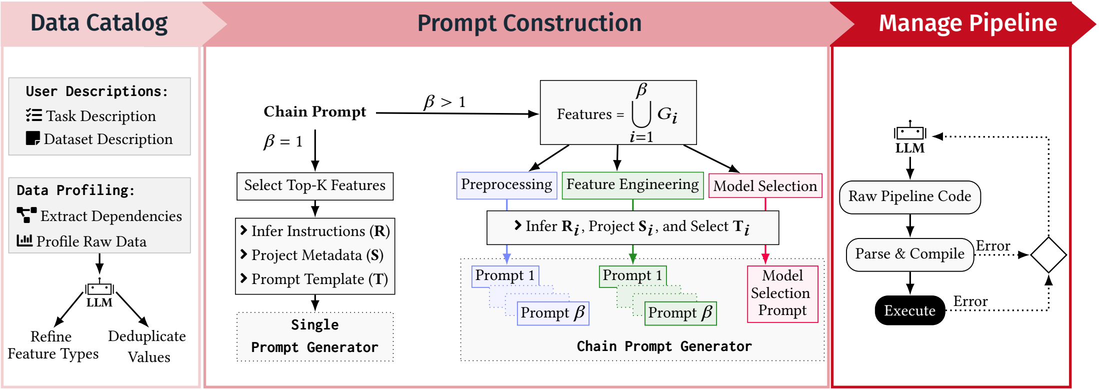
Data-centric ML pipeline generation in CatDB



Data-centric ML pipeline generation in CatDB



Data-centric ML pipeline generation in CatDB



CatDB-generated prompt & resulting pipeline

System Rules

Task: Generate a data science pipeline in Python 3.10.

Input: A dataset in CSV format, a schema that describes the columns and data types of the dataset, and a data profiling info that summarizes the statistics and quality of the dataset.

Output: A pipeline code that performs the following steps:

#1: Import the necessary libraries and modules.

#2: Load the training and test datasets. For the training data, utilize the variable "train_data.csv", and for the test data, employ the variable "test_data.csv".

#3: The user will provide the Schema, and Data Profiling Info of the dataset with columns appropriately named as attributes.

#4: Perform missing value imputation for features 'Address' and 'Zip'.

#5: Perform feature extraction (dataset contains categorical values).

#6: Perform feature selection (redundant columns e.g., 'Address' and 'Zip').

#7: Select an appropriate ML algorithm.

#8: Assign a default value to a particular hyperparameter.

#9: Evaluate the model.

Catalog Data

Dataset Description: A The dataset was obtained from multiple sources, including surveys, job posting sites, and other publicly available sources.A total of 6704 data points were collected.The dataset included fivevariables: age, experience, job role, and education level and salary.

Schema, and Data Profiling Info:

Experience (string), distinct [60%], missing [0%], categorical-vals [1 year, 2 years, 3 years]
Gender (string), distinct [40%], missing [0%], categorical-vals [Male, Female]
Address (string), distinct [40%], missing [20%], categorical-vals [CA, TX]
Zip (string), distinct [40%], missing [40%], categorical-vals [7050, 7871]
Salary (int, **target feature**), min-max vals [100, 310], total-vals [5]

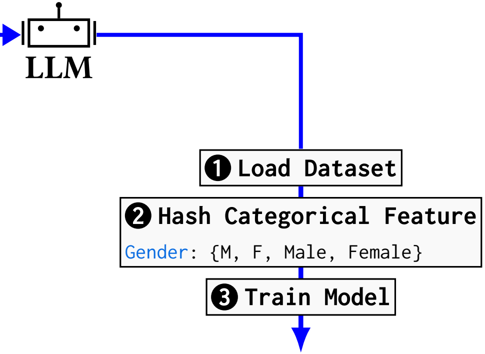
Categorical Features: Experience, Gender, Address, Zip

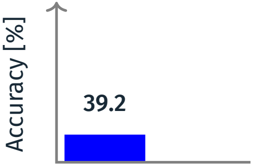
```python
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

train_data = pd.read_csv("train_data.csv")
test_data = pd.read_csv("test_data.csv")

categorical_features = ["Experience", "Gender", "Address", "Zip"]
categorical_transformer = Pipeline(steps=[
        ("imputer", SimpleImputer(strategy="most_frequent")),
        ("onehot", OneHotEncoder(handle_unknown="ignore"))])
preprocessor = ColumnTransformer(
        transformers = [("cat", categorical_transformer, categorical_features)],
        remainder="drop" )

model = RandomForestRegressor(n_estimators=100, random_state=42)
pipeline = Pipeline(steps=[ ("preprocessor", preprocessor), ("model", model)])

X_train = train_data.drop("Salary", axis=1)
y_train = train_data["Salary"]
X_test = test_data.drop("Salary", axis=1)
y_test = test_data["Salary"]

pipeline.fit(X_train, y_train)
y_test_pred = pipeline.predict(X_test)

test_r2 = r2_score(y_test, y_test_pred)
print(f"Test R^2: {test_r2}")
```

Example: Pipeline DAG in SOTA vs. CatDB

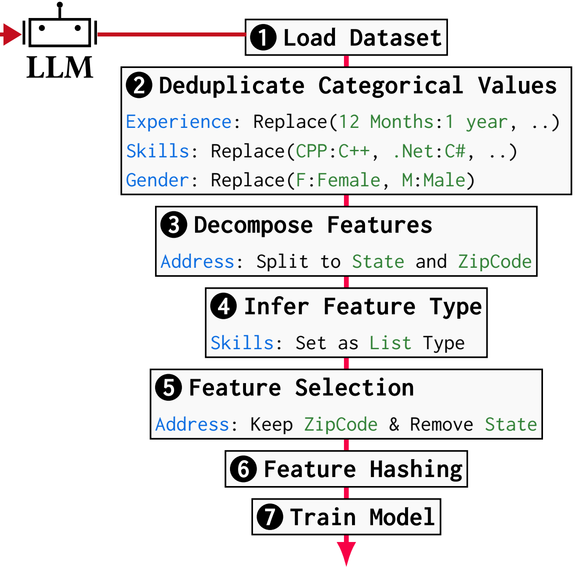
Salary dataset: comma-separated CSV file
Target feature: "**Salary**"

Experience	**Skills**	**Gender**	**Address**	**Salary**
12 Months	SQL,Java	F	7050 CA	100
two years	JavaScript	Female	TX 7871	150
36 months	C/C++,.Net	M	Texas	300
3 Years	JS,CPP,SQL	Female	7871	310
one Year	Python	Male	CA	200
2 Years	C#,Java	0	TX	175



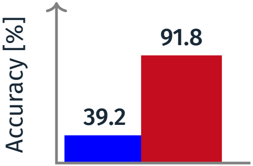


Example: Pipeline DAG in SOTA vs. CatDB



Salary dataset: comma-separated CSV file
Target feature: "**Salary**"

Experience	Skills	Gender	Address	Salary
12 Months	SQL,Java	F	7050 CA	100
two years	JavaScript	Female	TX 7871	150
36 months	C/C++,.Net	M	Texas	300
3 Years	JS,CPP,SQL	Female	7871	310
one Year	Python	Male	CA	200
2 Years	C#,Java	0	TX	175



Example of Data Catalog Update and Data Cleaning

Raw Dataset

Salary dataset: comma-separated CSV file Target feature: "**Salary**"				
Experience	**Skills**	**Gender**	**Address**	**Salary**
12 Months	SQL,Java	F	7050 CA	100
two years	JavaScript	Female	TX 7871	150
36 months	C/C++,.Net	M	Texas	300
3 Years	JS,CPP,SQL	Female	7871	310
one Year	Python	Male	CA	200
2 Years	C#,Java	0	TX	175

Example of Data Catalog Update and Data Cleaning

Raw Dataset

Salary dataset: comma-separated CSV file
Target feature: "**Salary**"

Experience	Skills	Gender	Address	Salary
12 Months	SQL,Java	F	7050 CA	100
two years	JavaScript	Female	TX 7871	150
36 months	C/C++,.Net	M	Texas	300
3 Years	JS,CPP,SQL	Female	7871	310
one Year	Python	Male	CA	200
2 Years	C#,Java	0	TX	175

Refine Duplicates

Clean Dataset								
#	**Experience**	**Gender**	**State**	**Zip**	**C++**	$\cdots$	**Python**	**Salary**
1	1 year	Female	CA	7050	0	$\cdots$	0	100
2	2 years	Female	TX	7871	0	$\cdots$	0	150
3	3 years	Male	TX		1	$\cdots$	0	300
4	3 years	Female		7871	1	$\cdots$	0	310
5	1 year	Male	CA		0	$\cdots$	1	200

Example of Data Catalog Update and Data Cleaning

Raw Dataset

Salary dataset: comma-separated CSV file Target feature: "**Salary**"				
Experience	**Skills**	**Gender**	**Address**	**Salary**
12 Months	SQL,Java	F	7050 CA	100
two years	JavaScript	Female	TX 7871	150
36 months	C/C++,.Net	M	Texas	300
3 Years	JS,CPP,SQL	Female	7871	310
one Year	Python	Male	CA	200
2 Years	C#,Java	0	TX	175

Refine Duplicates

Clean Dataset								
#	**Experience**	**Gender**	**State**	**Zip**	**C++**	**···**	**Python**	**Salary**
1	**1 year**	Female	CA	7050	0	···	0	100
2	**2 years**	Female	TX	7871	0	···	0	150
3	**3 years**	Male	TX		1	···	0	300
4	**3 years**	Female		7871	1	···	0	310
5	**1 year**	Male	CA		0	···	1	200

Update
Data Catalog

Data Catalog

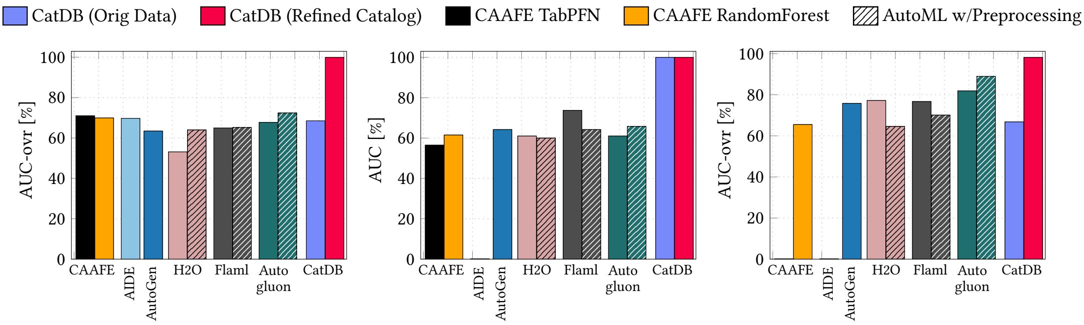
Column Name	**% Distinct**	**Feature Type**	**Samples**
Experience	**100**	**Sentence**	[**12 Months, two years, ...**]
Skills	100	Sentence	["Python,Java", ...]
Gender	60	Categorical	[F, Female, M]
Address	100	Sentence	[7050 CA, TX 7871, CA, ...]
Experience	**60**	**Categorical**	[**1 year, 2 years, 3 years**]
Skills	--	List	[SQL, Java, C++, ...]
Gender	40	Categorical	[Male, Female]
State	40	Categorical	[CA, TX]
Zip	40	Categorical	[7050, 7871]

Experiment Setup

- **Real-world Datasets (20 datasets):**
  - **single/multiple tables**,
  - **few/many samples**,
  - **small/large number of features**, and
  - **clean/dirty data**.

- **Comparing Systems:**
  - **LLM-based Systems:** CAAFE, AIDE, AutoGen
  - **AutoML Tools:** H2O, Flaml, Auto-Sklearn, AutoGluon
  - **AutoML-based Workflows:** Data cleaning w/ **SAGA** and **Learn2Clean**, and data augmentation w/ **ADASYN**

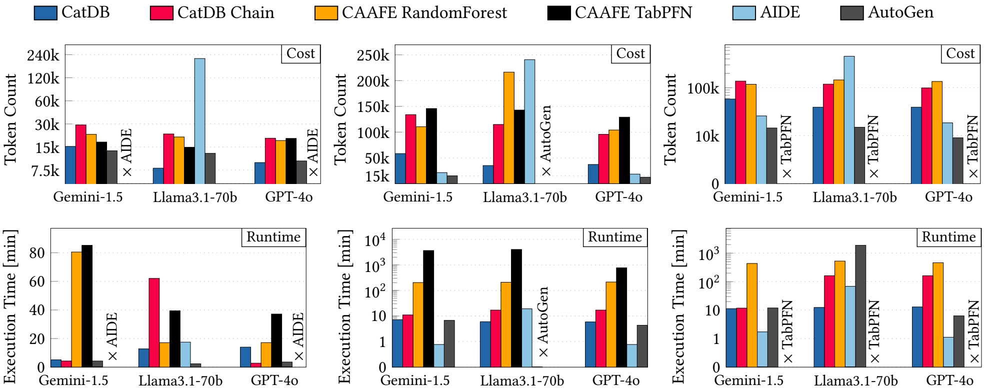
Performance Comparison (LLM = Gemini-1.5)



(a) Etailing (b) Wifi (c) Yelp

- **Deduplication ➜** Removes duplicates, balances labels.
- **Categorical Features ➜** Fixes formatting, transforms complexity.
- **Feature Refinement ➜** Drops constant/misread features, preserves distribution.

Cost and Runtime Comparison (10 iterations)

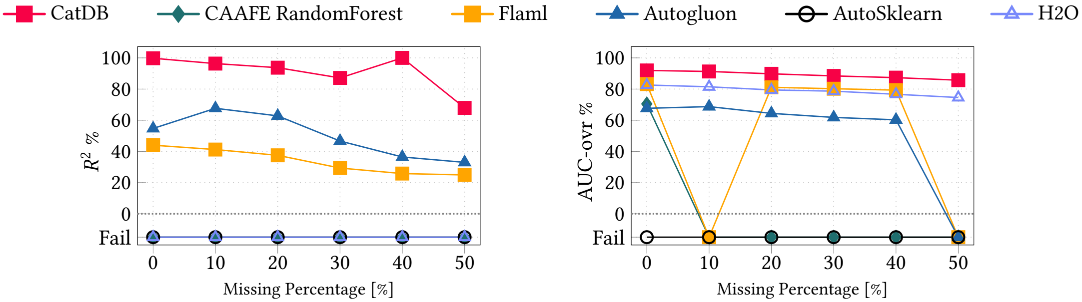


(a) Diabetes (b) Gas-Drift (c) Volkert

■ **Cost Efficiency:**
- AIDE & AutoGen use Hank-crafted prompts.
- CatDB consumes fewer tokens by projecting the Data Catalog.

■ **Runtime Speedup:**
- CatDB achieve **8x-14x** faster.
- AIDE & AutoGen consider only execution time.

Outlier and Missing Value Injection (Gemini-1.5)



(a) Utility MV + Outlier=5%

(b) Volkert MV + Outlier=5%

- **Robustness ➜** CatDB maintains high performance even with increasing missing values and 5% outliers.

Conclusions

- **Data Catalog Integration ➜** Use metadata & rules for tailoring pipelines.
- **Catalog Refinements ➜** Enhance catalogs to guide ML pipeline creation.
- **Prompt Chaining ➜** Sequence prompts to optimize generation.
- **Error Handling ➜** Validate, fix with knowledge base for reliable pipelines.

CatDB: Data-catalog-guided, LLM-based Generation of Data-centric ML Pipelines

Saeed Fathollahzadeh
Concordia University
saeed.fathollahzadeh@concordia.ca

Essam Mansour
Concordia University
essam.mansour@concordia.ca

Matthias Boehm
Technische Universität Berlin
matthias.boehm@tu-berlin.de