CROWN-102

Corporate Innovation Lab

2025

OVERTURE MAPS FOUNDATION

PROJECT B PLACES ATTRRIBUTES CONFLATION

Shreya Handa and Sriya Ramachandruni

START

MORE

ALL ABOUT PROJECT B

When a single real-world place has data from multiple sources, attributes like names and addresses vary. How do we know which attributes are more accurate?

SAME RESTAURANT, DIFFERENT SOURCES

FOURSQUARE DATA

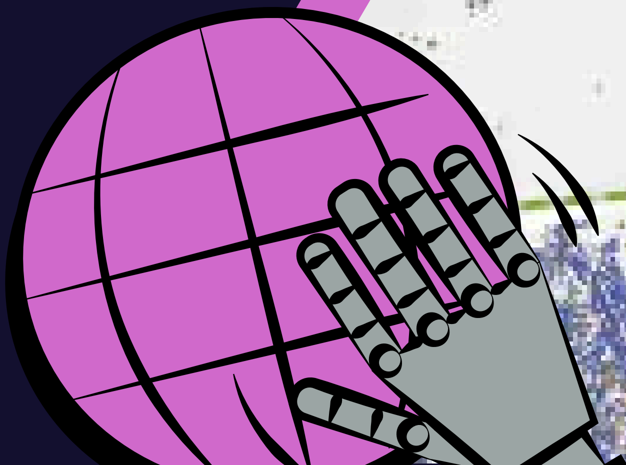
THE UPS STORE

1389 W 86TH ST
INDIANAPOLIS IN 46260

META DATA

THE UPS STORE 6285

1389 W 86TH ST
INDIANAPOLIS IN 46260 2101



THE OBJECTIVE

Develop automated decision logic to select the best attributes from conflicting sources, solving place-attribute conflation and supporting global access to reliable map data

INITIAL OKR'S

Objective 1: Build a "Golden Dataset" as Ground-Truth Proxy

Key Results 1:

- Look through at least 500-1,000 places from the given Yelp and OMF data
- Find and label at least 500 conflicting places, which will serve as a ground-truth proxy dataset.
- Achieve >90% consistency between ground-truth proxy dataset and information found online (Google, business websites, etc.).

Objective 2: Develop a rule-based system to compare with the ground-truth proxy dataset

Key Results 2:

- Highlight 3-5 logic-based rules (if missing, use others; choose the latest update,...).
- Implement in code using Python & SQL using these rules.
- Achieve >80% accuracy of the total number of correct attributes matching ground truth.

Objective 3: Build a machine learning model to automate the selection of place attributes

Key Results 3:

- Train at least 2-3 ML approaches (e.g., Random Forest, XGBoost, simple neural network) on the ground-truth proxy dataset.
- Achieve > 85% accuracy on held-out test set (5% improvement over rule-based baseline)
- Compare ML vs. rule-based performance across multiple metrics
- Document features importance to understand which signals drive ML decisions.

Objective 4: Testing and Evaluation

Key Results 4:

- Test best-performing algorithm (ML or rule-based) on 100-200 new, unseen places not in training/test sets.
- Achieve > 80% accuracy on this final validation set.
- Conduct error analysis identifying failure patterns and edge cases.
- Create deployment documentation

FINAL OKR'S

Objective: Develop rule-based and machine learning pipelines for resolving place-attribute conflation.

KR1 — Normalization Dataset

- Build and document a normalized dataset covering 20+ location patterns that are reused by both the rule-based and ML pipelines, and validate that it has <10% parsing errors.

KR2 — Rule-Based System Accuracy

- Implement 5 rules per attribute (name, phone, address, website, categories) and achieve ≥ 80% attribute-level accuracy on the rule-based conflation dataset.

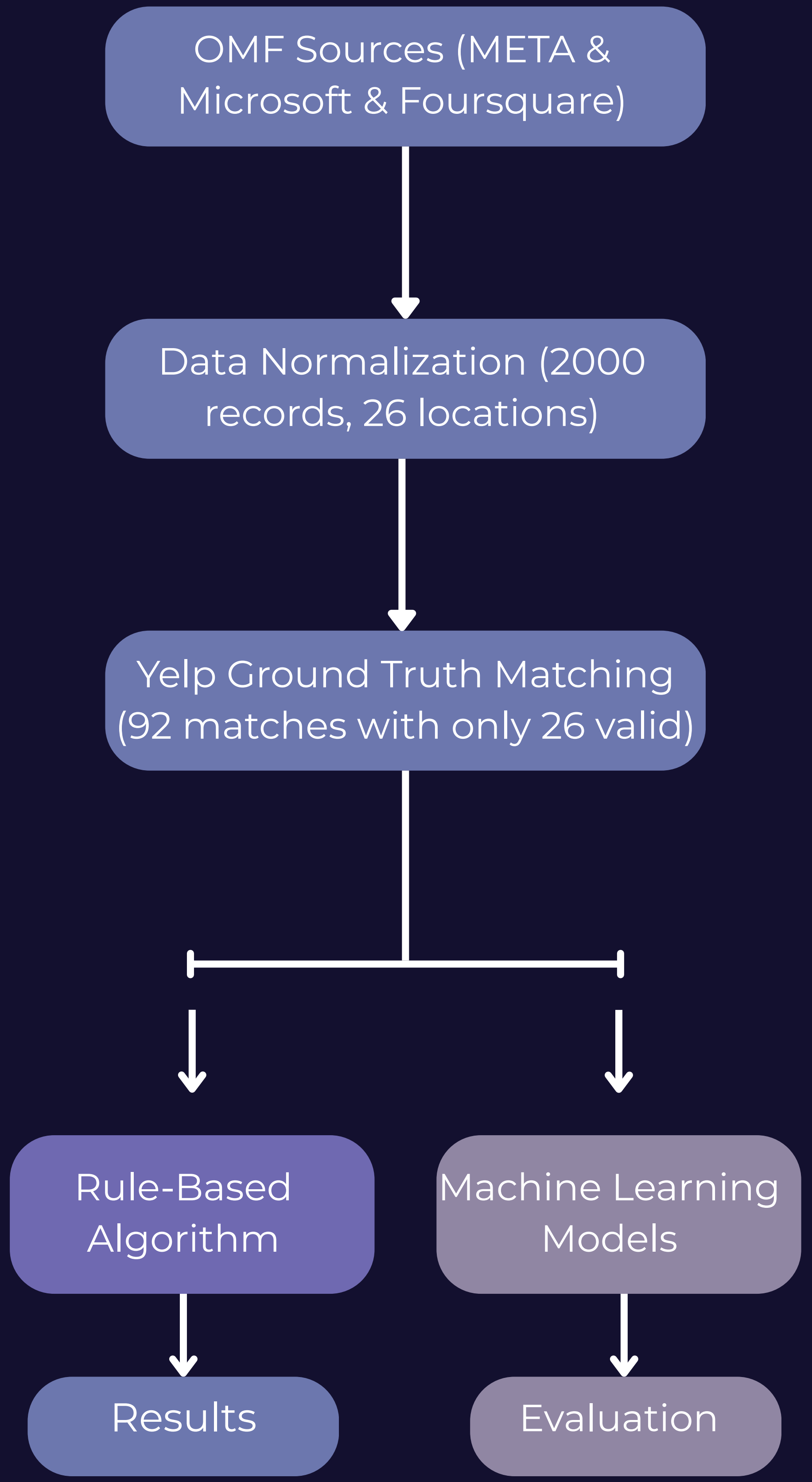
KR3 — Machine Learning Models

- Train 5 ML models (one per attribute) and achieve ≥ 80% normalized, attribute-based accuracy on the ML-based conflation dataset.

KR4 — Comparative Evaluation (Rule-Based vs ML)

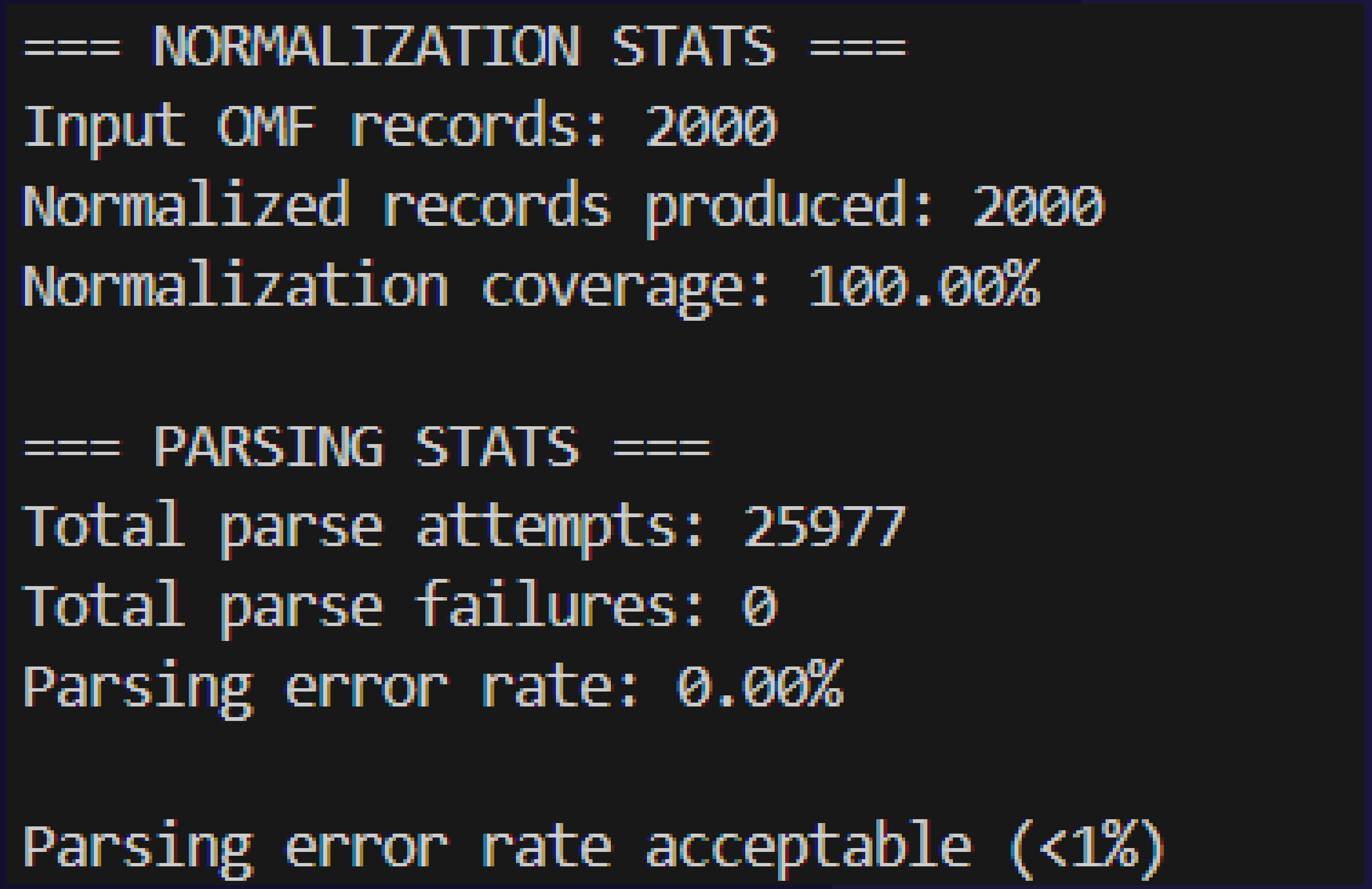
- By Week 10, deliver evaluation comparing rule-based and ML pipelines, identifying 1–2 strengths and 1–2 limitations considering long-term maintainability and explainability.

APPROACH


OMF Sources (META & Microsoft & Foursquare)
Data Normalization (2000 records, 26 locations)
Yelp Ground Truth Matching (92 matches with only 26 valid)
Rule-Based Algorithm
Machine Learning Models
Results
Evaluation

NORMALIZATION

```
NORMALIZATION PIPELINE
(OMF_NORMALIZE_DATA.PY)
├─ PARSE 2000 OMF RECORDS
├─ STANDARDIZE SCHEMA ACROSS SOURCES
├─ EXTRACT 5 KEY ATTRIBUTES PER PLACE
│  • NAME
│  • PHONE
│  • ADDRESS
│  • WEBSITE
│  • CATEGORIES
└─ OUTPUT: NORMALIZED_SOURCES.CSV
```



```
=== NORMALIZATION STATS ===
Input OMF records: 2000
Normalized records produced: 2000
Normalization coverage: 100.00%

=== PARSING STATS ===
Total parse attempts: 25977
Total parse failures: 0
Parsing error rate: 0.00%

Parsing error rate acceptable (<1%)
```

✓ Objective 1, KR 1 achieved

NORMALIZED EXAMPLE

place_id	source	record_id	update_time	name	categories	phone	website	socials	address	confidence
08f44f055a9a016e0390f050aa3c93c0	meta	4632734703927	2025-01-06T00:00:00.000Z	Goin' Postal Jacksonville	{"primary": "shipping_center", "alternate": ["freight_and_cargo_service", "post_office"]}	["+19049989600"]	["http://www.goinpostaljacksonville.com/"]	["https://www.facebook.com/463273470392736"]	[{"freeform": "7643 Gate Pkwy", "locality": "Jacksonville", "postcode": "32256-2892", "region": "FL", "country": "US"}]	0.9962614185921
08f44f055a9a016e0390f050aa3c93c0	msft	844424934663	2025-01-06T00:00:00.000Z	Goin' Postal Jacksonville	{"primary": "shipping_center", "alternate": ["freight_and_cargo_service", "post_office"]}	["+19049989600"]	["http://www.goinpostaljacksonville.com/"]	["https://www.facebook.com/463273470392736"]	[{"freeform": "7643 Gate Pkwy", "locality": "Jacksonville", "postcode": "32256-2892", "region": "FL", "country": "US"}]	0.9962614185921
08f44f055a9a016e0390f050aa3c93c0	msft	168884986566	2025-01-06T00:00:00.000Z	Goin' Postal Jacksonville	{"primary": "shipping_center", "alternate": ["freight_and_cargo_service", "post_office"]}	["+19049989600"]	["http://www.goinpostaljacksonville.com/"]	["https://www.facebook.com/463273470392736"]	[{"freeform": "7643 Gate Pkwy", "locality": "Jacksonville", "postcode": "32256-2892", "region": "FL", "country": "US"}]	0.9962614185921
08f44f055a9a016e0390f050aa3c93c0	msft	140737488661	2025-01-06T00:00:00.000Z	Goin' Postal Jacksonville	{"primary": "shipping_center", "alternate": ["freight_and_cargo_service", "post_office"]}	["+19049989600"]	["http://www.goinpostaljacksonville.com/"]	["https://www.facebook.com/463273470392736"]	[{"freeform": "7643 Gate Pkwy", "locality": "Jacksonville", "postcode": "32256-2892", "region": "FL", "country": "US"}]	0.9962614185921
08f44f055a9a016e0390f050aa3c93c0	msft	140737488508	2025-01-06T00:00:00.000Z	Goin' Postal Jacksonville	{"primary": "shipping_center", "alternate": ["freight_and_cargo_service", "post_office"]}	["+19049989600"]	["http://www.goinpostaljacksonville.com/"]	["https://www.facebook.com/463273470392736"]	[{"freeform": "7643 Gate Pkwy", "locality": "Jacksonville", "postcode": "32256-2892", "region": "FL", "country": "US"}]	0.9962614185921
08f44f055a9a016e0390f050aa3c93c0	msft	112589990937	2025-01-06T00:00:00.000Z	Goin' Postal Jacksonville	{"primary": "shipping_center", "alternate": ["freight_and_cargo_service", "post_office"]}	["+19049989600"]	["http://www.goinpostaljacksonville.com/"]	["https://www.facebook.com/463273470392736"]	[{"freeform": "7643 Gate Pkwy", "locality": "Jacksonville", "postcode": "32256-2892", "region": "FL", "country": "US"}]	0.9962614185921
08f44f055a9a016e0390f050aa3c93c0	microsoft_structured	168884986566	2023-05-16T17:32:11.307	Goin' Postal Jacksonville	{"primary": "vehicle_shipping", "alternate": ["courier_and_delivery_services", "mailbox_center", "shipping_center", "post_office", "automotive"]}	["9049989600"]	["https://www.goinpostaljacksonville.com/"]		[{"freeform": "7643 Gate Pkwy Ste 104", "locality": "Jacksonville", "region": "FL", "country": "US", "postcode": "32256"}]	0.

RULE-BASED CONFLATION

1. NAME ATTRIBUTE PARAMETERS

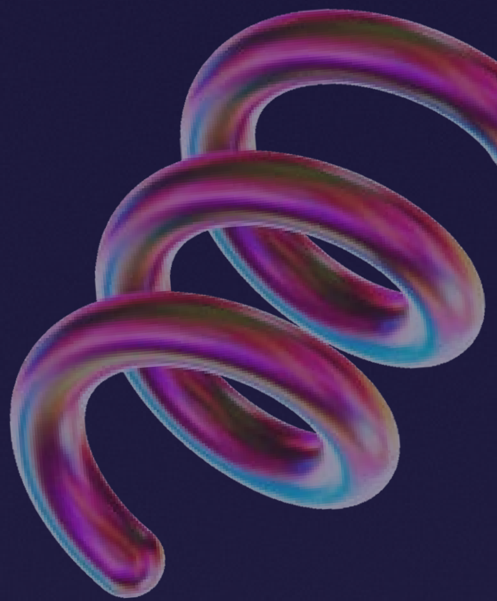
2. ADDRESS ATTRIBUTE PARAMETERS

3. PHONE ATTRIBUTE PARAMETERS

4. WEBSITE ATTRIBUTE PARAMETERS

5. CATEGORIES ATTRIBUTE PARAMETERS

NAME ATTRIBUTE PARAMETERS

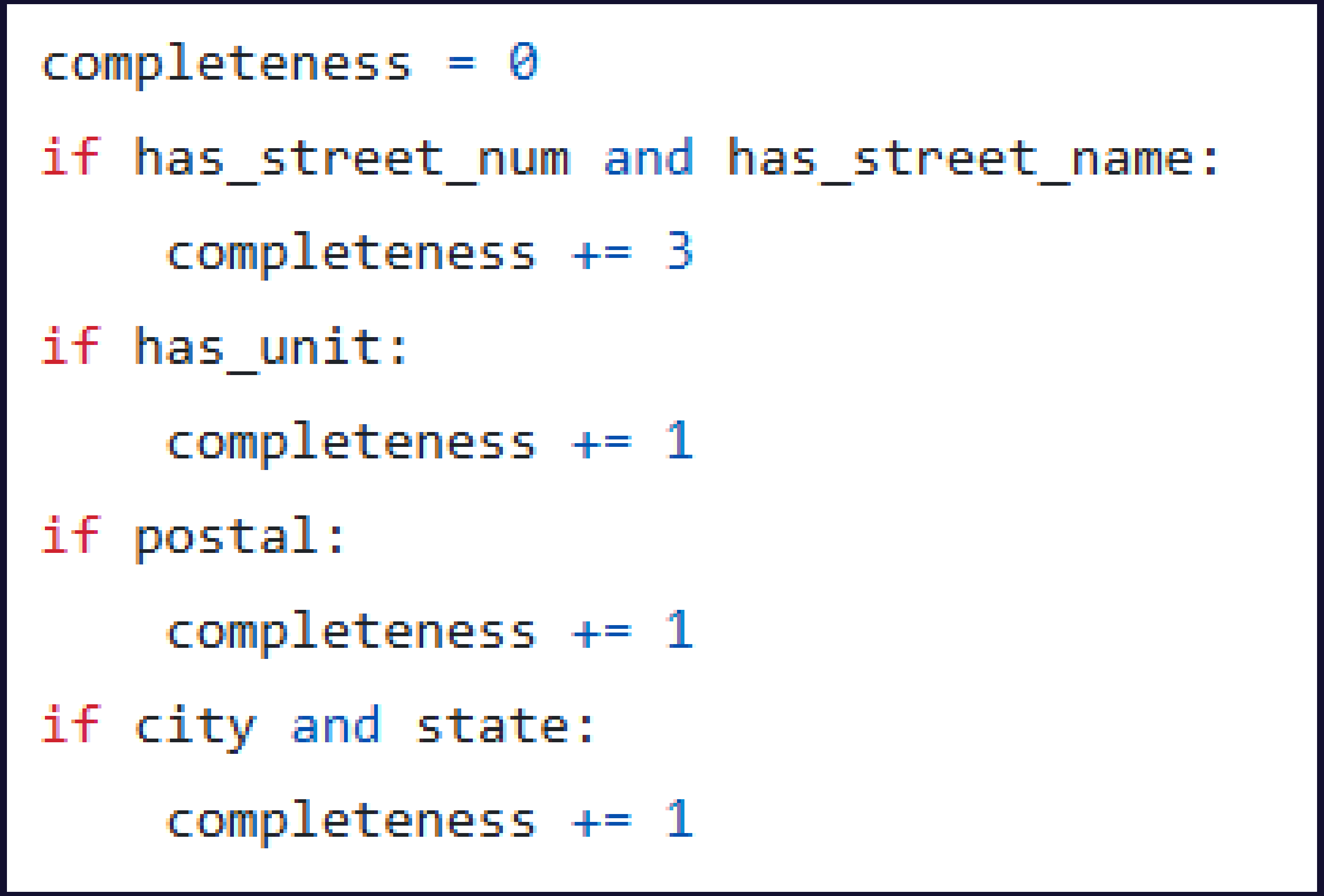


1. **Clean the name and detect noisy names**
    a. Remove punctuation
    b. Penalize names that
        i. Look like sentences
        ii. Contain “hours”, “address”, etc.
        iii. Remove store numbers (store #1482, 0034, etc.) (only if other clean names exist)
2. **Compare similarity across providers**
    a. Compute pairwise similarity of all sources
    b. If has high similarity to others, that name is most likely the truth
3. **Final choice**
    a. Pick the name with the highest similarity to others
    b. Has the cleanest brand name, less noise
    c. If still tied: Choose name amongst best provider (Meta > Microsoft > Foursquare)

ADDRESS ATTRIBUTE PARAMETERS

1. **Parse all address components**
   a. Extract: Street, City, State, Postal code, Country
2. **Create a "match key" = street + city + state**
   a. If multiple providers give the same key, they refer to the same physical place
3. **Majority wins**
   a. If all sources have the same key, that address form is more trustworthy
4. **Within winners, choose the most complete**
   a. +3 = street number + street name
   b. +1 = unit/apartment
   c. +1 = postal
   d. +1 = city/state

The address with the most agreement, highest completeness, and longest full address (more specific) wins.



```
completeness = 0
if has_street_num and has_street_name:
    completeness += 3
if has_unit:
    completeness += 1
if postal:
    completeness += 1
if city and state:
    completeness += 1
```

PHONE ATTRIBUTE PARAMETERS

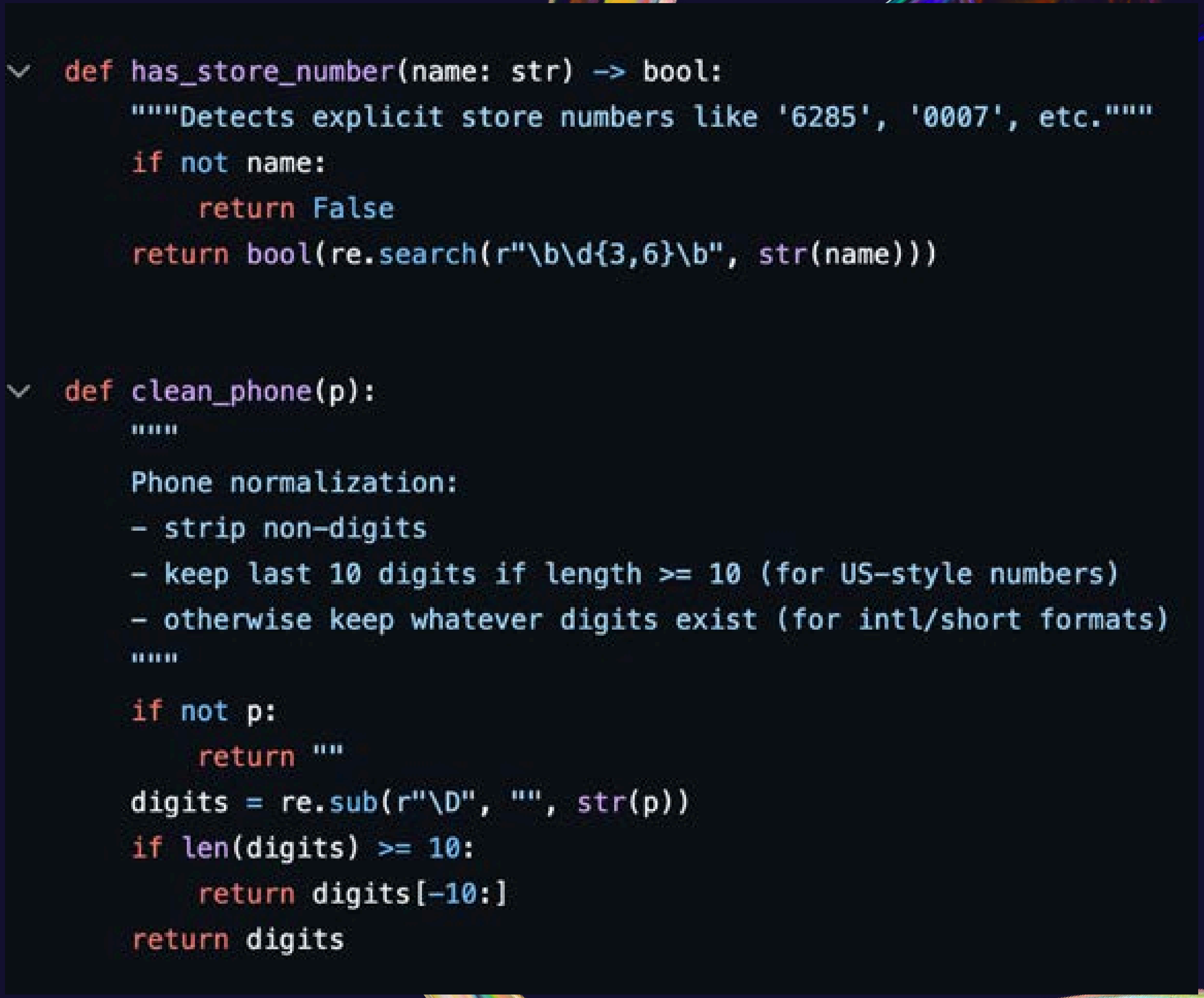
Normalization:

- Strip all non-digits
- Keep last 10 digits if length ≥ 10 (US format)
- Otherwise keep all available digits (international)

Selection Logic:

- Majority vote: If ≥2 sources agree on same number, choose that number

Fallback: If all numbers differ, choose the number from Pick from highest-priority source (Meta > Microsoft > Foursquare)



```python
def has_store_number(name: str) -> bool:
    """Detects explicit store numbers like '6285', '0007', etc."""
    if not name:
        return False
    return bool(re.search(r"\b\d{3,6}\b", str(name)))


def clean_phone(p):
    """
    Phone normalization:
    - strip non-digits
    - keep last 10 digits if length >= 10 (for US-style numbers)
    - otherwise keep whatever digits exist (for intl/short formats)
    """
    if not p:
        return ""
    digits = re.sub(r"\D", "", str(p))
    if len(digits) >= 10:
        return digits[-10:]
    return digits
```

WEBSITE ATTRIBUTE PARAMETERS

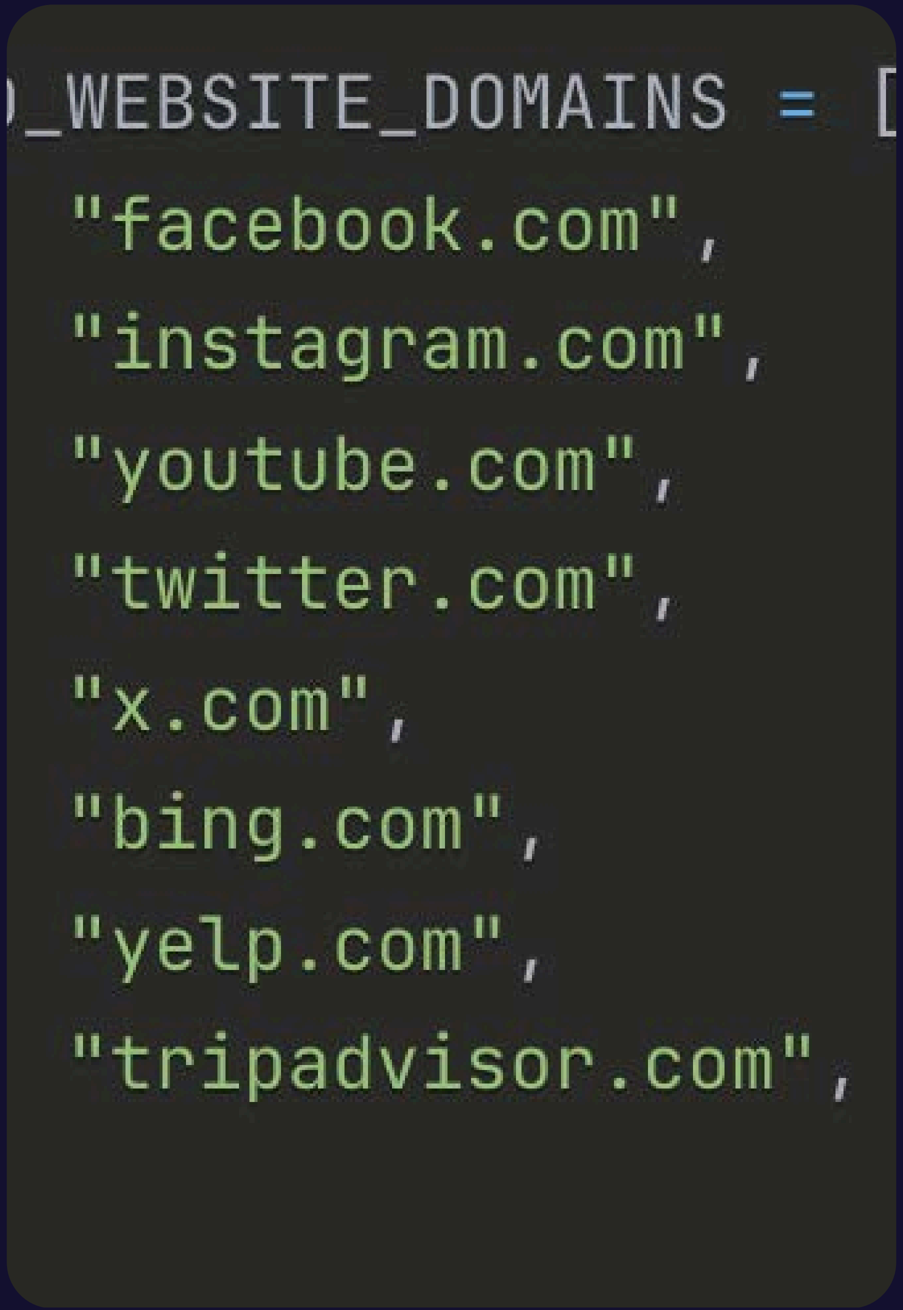
1. **Normalization:**

Reduce URL → domain

- https://www.starbucks.com/store/1234 → starbucks.com

2. Reject junk domains:

Reject "junk domains" like facebook.com, instragram.com, yelp.com, tripadvisor.com ONLY if a real website exists.



```
_WEBSITE_DOMAINS = [
  "facebook.com",
  "instagram.com",
  "youtube.com",
  "twitter.com",
  "x.com",
  "bing.com",
  "yelp.com",
  "tripadvisor.com",
```

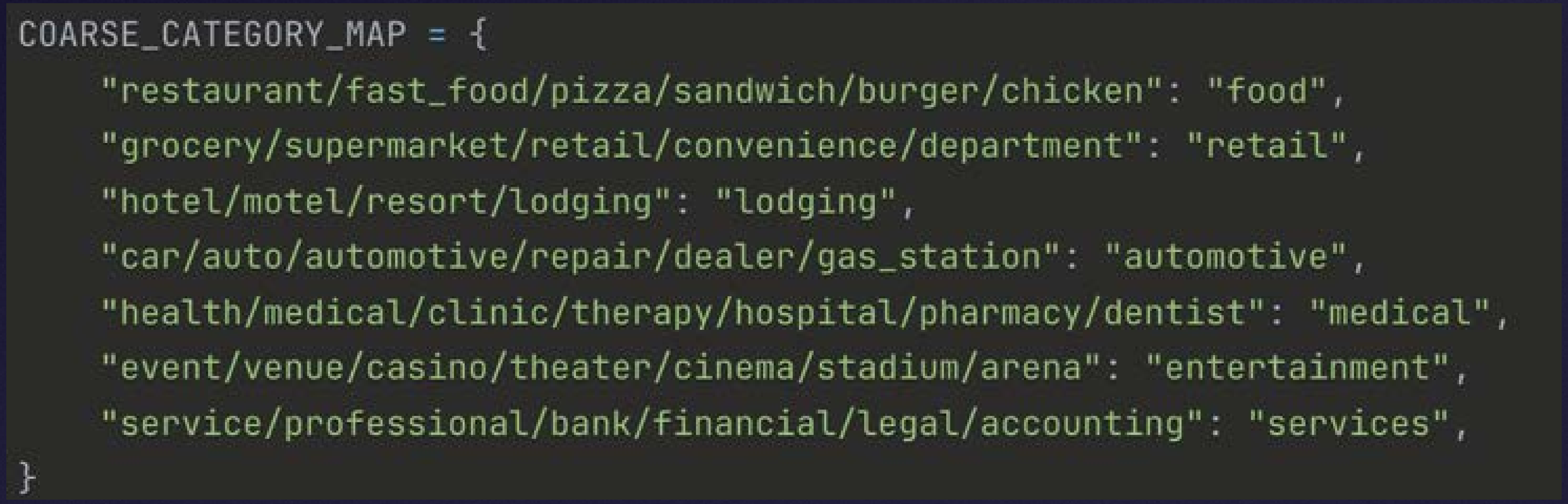
3. Selection Logic:

Pick the domain that appears the most frequently across all URLs and all sources.

4. Among domains, choose:

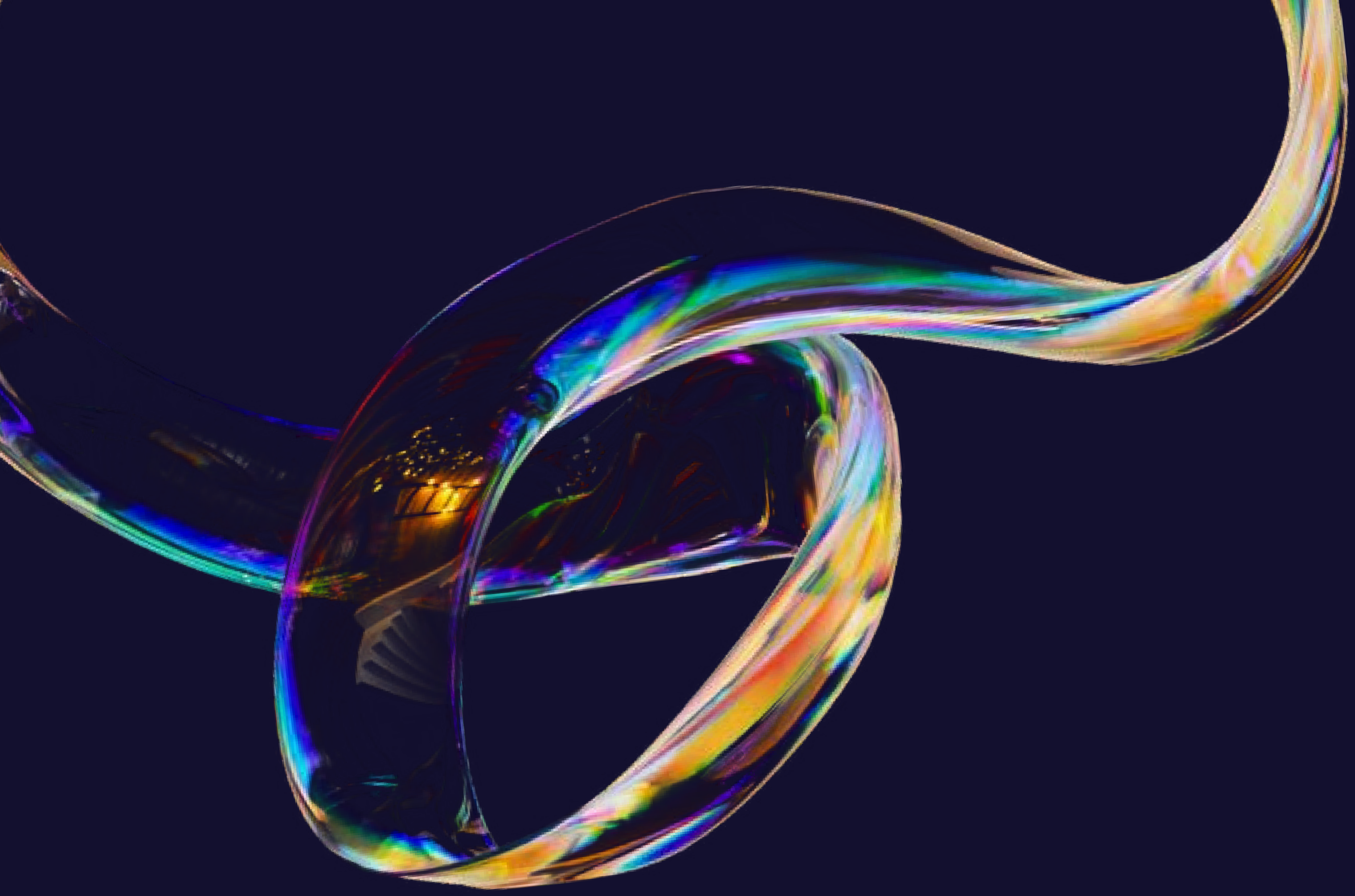
- Higher brand_match, meaning websites that contain name tokens ("starbucks" in "starbucks.com")
- Shorter URL length as it has less tracking

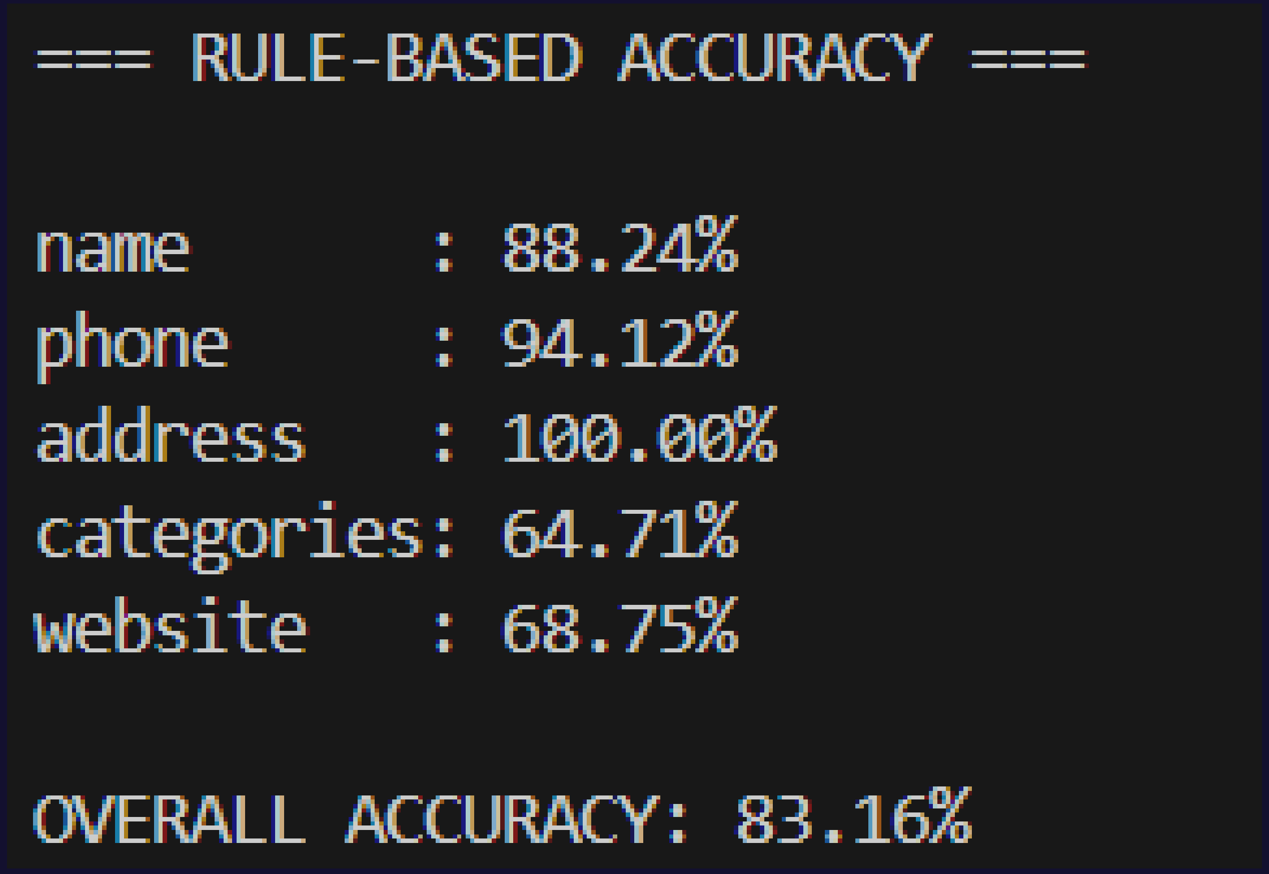
CATEGORIES ATTRIBUTE PARAMETERS



```
COARSE_CATEGORY_MAP = {
    "restaurant/fast_food/pizza/sandwich/burger/chicken": "food",
    "grocery/supermarket/retail/convenience/department": "retail",
    "hotel/motel/resort/lodging": "lodging",
    "car/auto/automotive/repair/dealer/gas_station": "automotive",
    "health/medical/clinic/therapy/hospital/pharmacy/dentist": "medical",
    "event/venue/casino/theater/cinema/stadium/arena": "entertainment",
    "service/professional/bank/financial/legal/accounting": "services",
}
```

1. Normalize all categories
2. Map each normalized category into a coarse bucket dictionary
   a. "pizza_restaurant" → Food
   b. "clinic" → Medical
   c. "supermarket" → Retail
3. If majority of sources map to "Food", then the winning course bucket is "Food"
4. Choose the most specific category label within the winning bucket looking at:
   a. Number of words (higher specificity)
      i. "Chicken restaurant" > "restaurant"
   b. Length of label (longer text, more specific)





```
=== RULE-BASED ACCURACY ===

name      : 88.24%
phone     : 94.12%
address   : 100.00%
categories: 64.71%
website   : 68.75%

OVERALL ACCURACY: 83.16%
```

RULE-BASED EVALUATION

1. Created a Golden Dataset
   a. Matched 26 valid locations between Yelp and OMF data
   b. We manually determined the truth attribute-value for each location using Yelp as our ground truth. This allowed us to come up with a set of rules we used in our rule-based algorithm
2. Ran our rule-based algorithm on 2,000+ OMF places
3. Compared the output of our rule-based algorithm to the truth attribtue-values in our Golden Dataset, calculating the final overall accuracy

✓ Objective 1, KR 2 achieved

MACHINE LEARNING CONFLATION

The Challenge: Rule-based algorithms are powerful, but they have limitations:

- Fixed thresholds don't adapt to data patterns
- Manual tuning required for new edge cases
- Can't capture complex multi-factor relationships

The ML Solution: What if the algorithm could learn from examples?

Our Approach:

- Show the algorithm 26 examples of "correct" choices
- Let it discover patterns automatically
- Apply learned patterns to predict best source for new places

Key Question: Can machine learning outperform explicit rules by learning from data? Even if iit does, is it more maintainable in the long-run?

Model Type: Logistic Regression & Random Forest Classifier
Training Data: 26 labeled places from VALID_MATCHES.csv
Models Trained: 5 independent models (one per attribute)

PHASE 1: FEATURE ENGINEERING

For EACH of the 5 attributes (name, phone, address, website, categories):

1. Get ground truth-attribute values for each location from Yelp, creating Golden Dataset
2. Calculate 3 features PER SOURCE (9 features total):
    - Feature 1: Similarity Score ({SOURCE}_sim)
    - Feature 2: Exact Match ({SOURCE}_exact)
    - Feature 3: Data Present ({SOURCE}_present)

Input Files:

- **NORMALIZED_SOURCES.csv - All OMF data (5,669 records with META, MSFT, Foursquare)**
- **ML_GOLDEN_DATASET_TEMPLATE.csv - 26 manually labeled places with ground truth from Yelp**

OUTPUT FILES

- `ML_TRAIN_FEATURES_name.csv`
- `ML_TRAIN_FEATURES_phone.csv`
- `ML_TRAIN_FEATURES_address.csv`
- `ML_TRAIN_FEATURES_website.csv`
- `ML_TRAIN_FEATURES_categories.csv`

foursquare_sim	foursquare_exact	foursquare_present	meta_sim	meta_exact	meta_present	microsoft_sim	microsoft_exact	microsoft_present
0	0	0	5	1	1	5	1	1
4	1	1	4	1	1	0	0	0
0	0	0	3	1	1	3	1	1
0	0	0	5	1	1	5	1	1

PHASE 2: MODEL TRAINING

Purpose: Teach the algorithm to recognize patterns that indicate which source is most accurate

For EACH attribute (name, phone, address, website, categories), we train 2 models-- Logistic Regression and Random Forest and save the better of the two models :

- Creates decision trees
- Each tree learns different patterns from the data
- Trees vote on final prediction (hgiher accuracy wins)

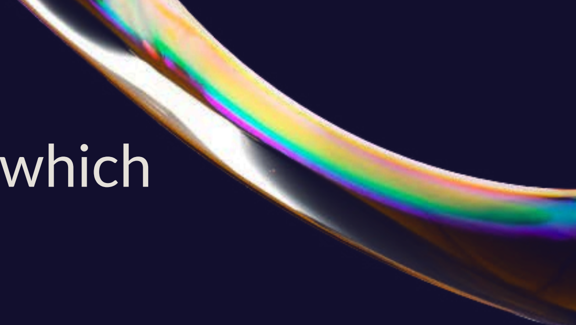
1. Load the training data

2. Encode labels as numbers

3. Train Linear Regression & Random Forest model

4. Save 5 trained models into **models/<ATTRIBUTE>_model.joblib**
   - Why train speerate models per attribute? Because different attributes have different patterns

These rules emerge naturally from the training data, we don't code them explicitly!



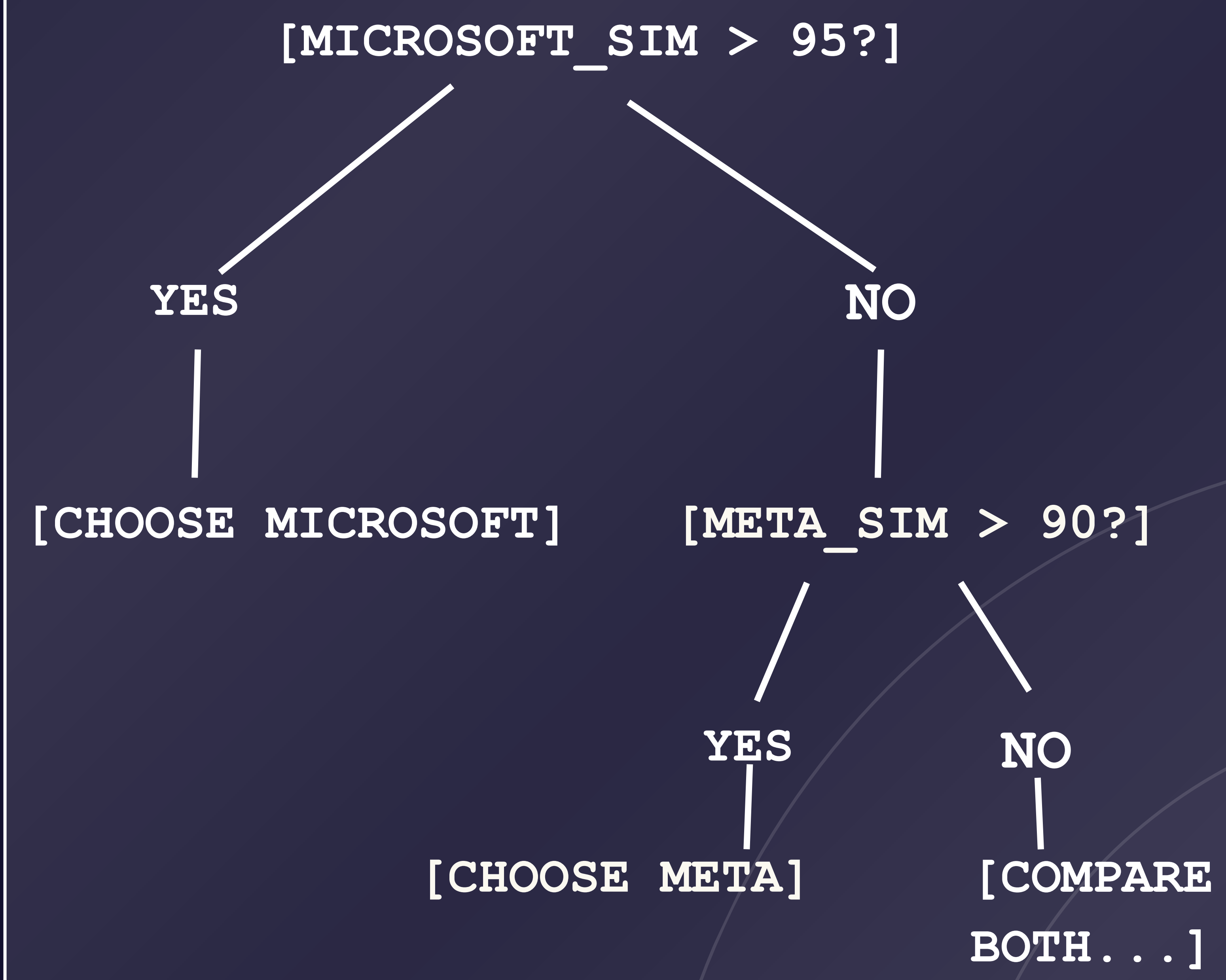
HOW DOES THE MODEL "LEARN"?

Training is pattern recognition

The model sees 92 examples like:
Example 1: [0, 0, 0, 87, 0, 1, 100, 1, 1] → Answer: microsoft
Example 2: [0, 0, 0, 95, 1, 1, 85, 0, 1] → Answer: meta
Example 3: [0, 0, 0, 70, 0, 1, 70, 0, 1] → Answer: meta .

After seeing many examples, it learns:

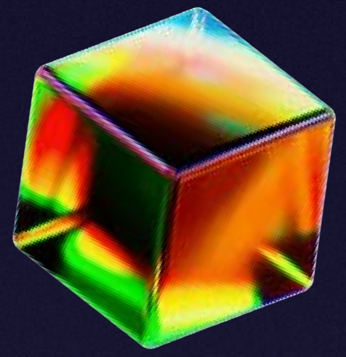
- "When microsoft_sim = 100 and exact = 1, almost always choose Microsoft"
- "When meta_sim > microsoft_sim by 10+ points, usually choose Meta"
- "When both similar, presence doesn't matter much"



THESE PATTERNS ARE ENCODED IN THE DECISION TREE STRUCTURES

PHASE 3: INFERENCE (PREDICTION)

Purpose: Use trained models to predict best attributes for NEW, unlabeled places, specifically 2,000+ OMF places. At inference time, we don't knwo the truth, so we approximate:



EXTRACT FEATURES (SAME WAY AS TRAINING)

LOAD TRAINED MODEL

MAKE PREDICTION

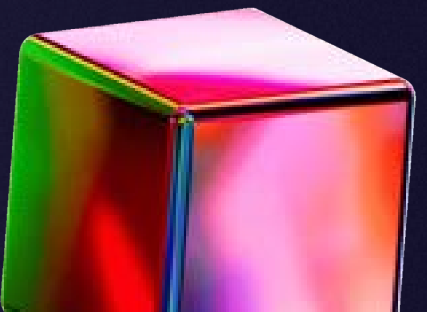
SELECT THE ATTRIBUTE VALUE FROM PREDICTED SOURCE

REPEAT FOR ALL 5 ATTRIBUTES

DETERMINE OVERALL BEST ATTRIBUTE (MAJORITY VOTE)

OUTPUT FILES

- `ML_INFER_FEATURES_name.csv`
- `ML_INFER_FEATURES_phone.csv`
- `ML_INFER_FEATURES_address.csv`
- `ML_INFER_FEATURES_website.csv`
- `ML_INFER_FEATURES_categories.csv`

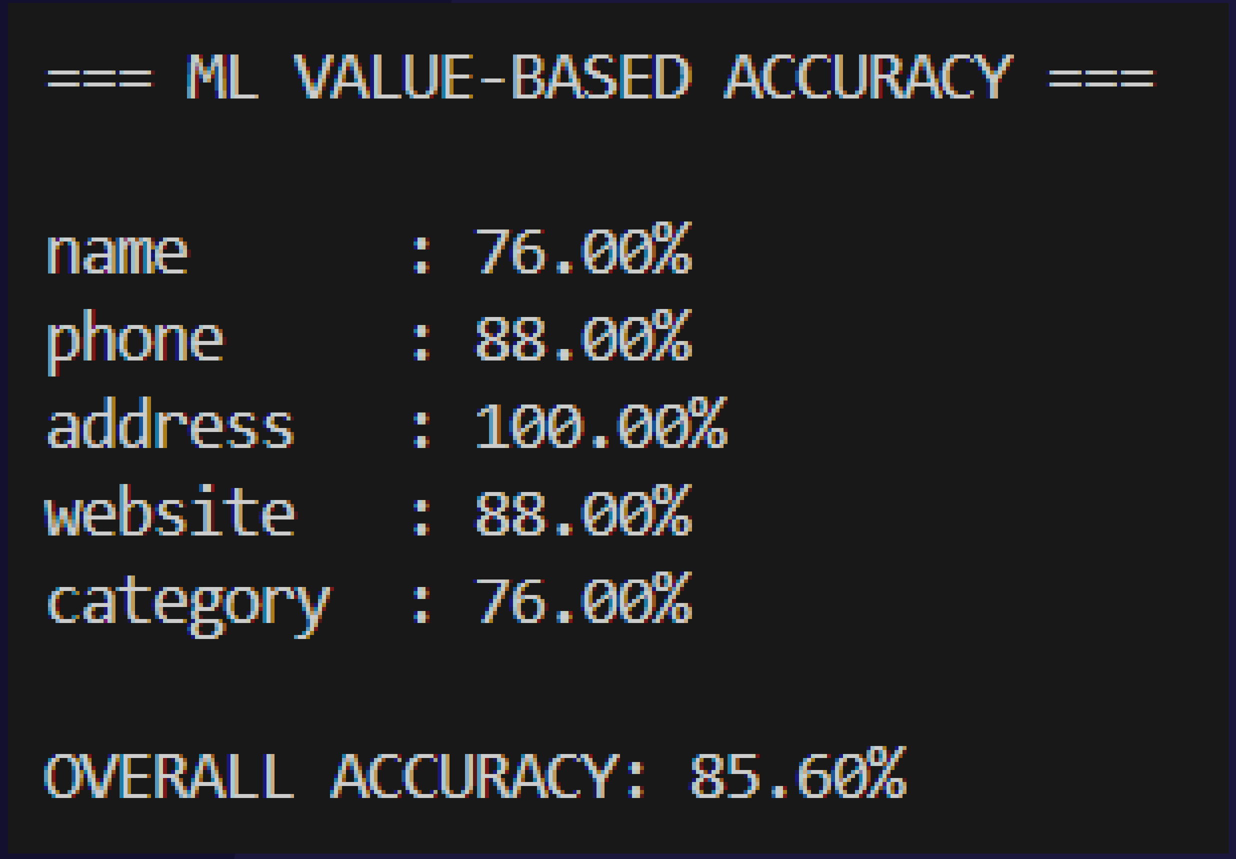


PHASE 4: EVALUATION

Measure how well ML predictions match ground truth.

Process:

1. Load ML predictions and Golden Dataset
2. Compare each attribute
3. Calculate overall accuracy



```
=== ML VALUE-BASED ACCURACY ===

name       : 76.00%
phone      : 88.00%
address    : 100.00%
website    : 88.00%
category   : 76.00%

OVERALL ACCURACY: 85.60%
```

✓ Objective 1, KR3 achieved

LIVE DEMO

project-terraforma/
Shreya-Sriya-Rule-...

2 Contributors 0 Issues 5 Stars 1 Fork

project-terraforma/Shreya-Sriya-Rule-Based-ML-Attacking-Places-Attribution-Conflation

Contribute to project-terraforma/Shreya-Sriya-Rule-Based-ML-Attacking-Places-Attribution-Conflation development by creating an account on GitHub.

GitHub

OUTCOMES

Both algorithms exceed ≥ 80% target accuracy

RULE-BASED

83.16% OVERALL

- ADDRESS: 100.00%
- PHONE: 94.12%
- NAME: 88.24%
- WEBSITE: 68.75%
- CATEGORIES: 64.71%

MACHINE LEARNING

85.60% OVERALL

- ADDRESS: 100.00%
- PHONE: 88.00%
- WEBSITE: 88.00%
- NAME: 76.00%
- CATEGORIES: 76.00%

RULE-BASED STRENGTHS

- Explainable - every decision traceable
- Deterministic and consistent
- Easy to debug and maintain
- Requires manual updates for new patterns
- Brittle to schema changes

ML STRENGTHS

- Learns complex patterns automatically
- Scales without manual rule expansion
- Slightly higher accuracy (85.6% vs 83.2%)
- Black box - less transparent decisions
- Requires larger training datasets

OVERTURE MAPS
FOUNDATION

FINAL REFLECTION AND THOUGHTS

Teamwork and Professional Work Environment

We learned a lot about the iterative process of receiving feedback and making change for project success. Throughout the quarter, we utilized OMF lead engineers, Albi and Felix, and Professor Rao to continuously refine our approach to better solve place-attribute conflation. This includes changing our OKR's to be more measurable and changing the way we normalized our data and created our Golden Dataset.

More Data the Merrier

We were only able to get 26 valid matches between OMF and Yelp data to train our rule-based algorithm and ML models. In the future, we would like repeat this project and train with a larger dataset as those additional locations will reveal new patterns otherwise missed. It will help increase the accuracy and reliability of our algorithms for when OMF integrates new map data in the future.

2025

HOORAY!

THANK YOU

END GAME