Enhancing Machine Learning and Data Visualization Pipelines with Isomorphisms

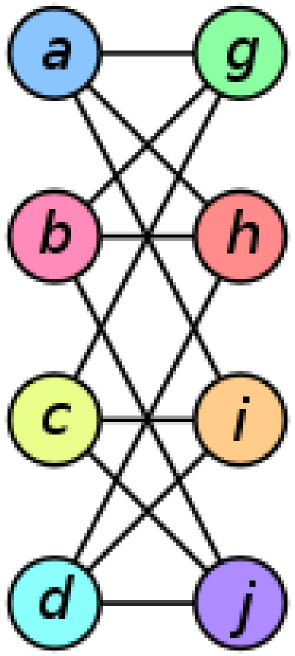
Jason A. Grafft
Knowledge Engineer, relationalAI
jason@grafft.co

Graph Isomorphism*

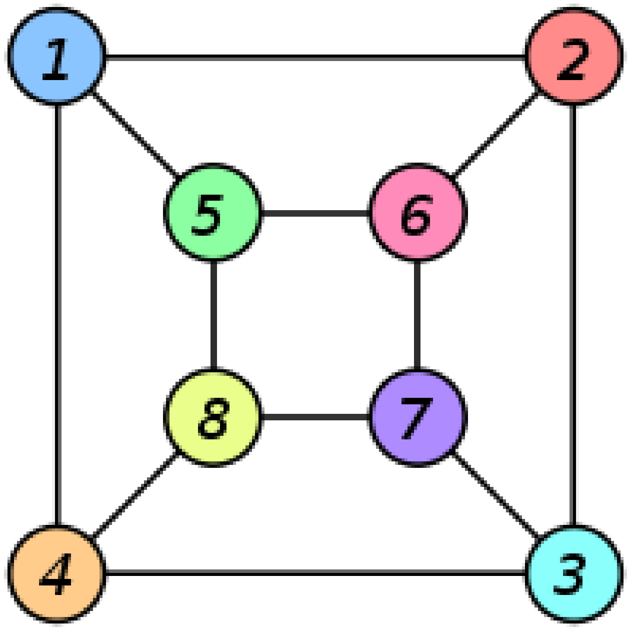
An **isomorphism of graphs** G and H is a bijection between the vertex sets of A and B

$$f : V(G) \to V(H)$$

Graph G



Graph H

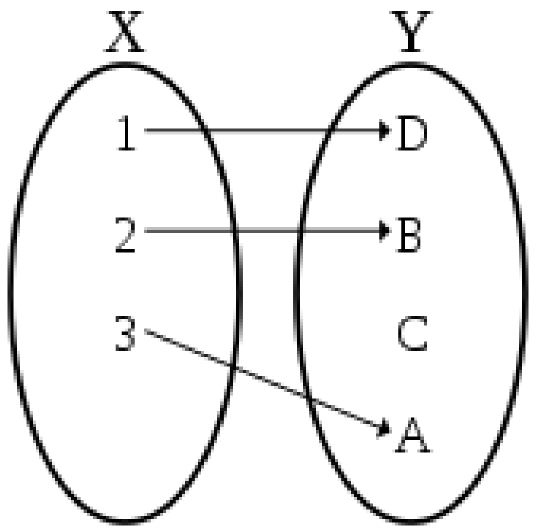


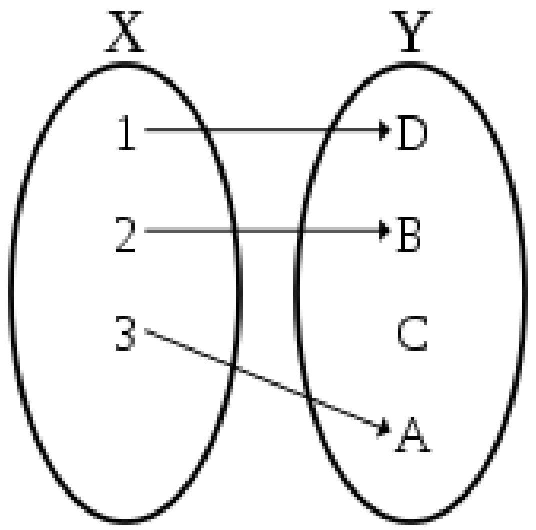
Isomorphism between G and H

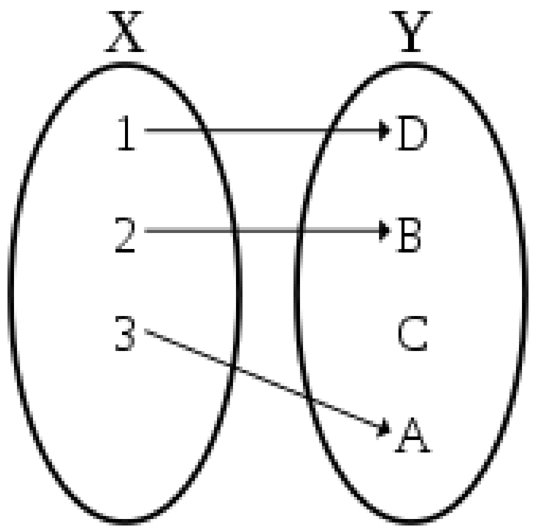
f(a) = 1
f(b) = 6
f(c) = 8
f(d) = 3
f(g) = 5
f(h) = 2
f(i) = 4
f(j) = 7

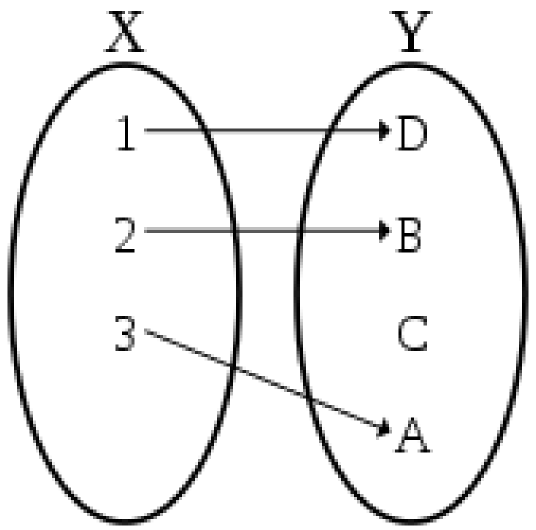
* - Definition and images from Wikipedia, https://en.wikipedia.org/wiki/Graph_isomorphism

Bijection*









`bijection?` $\perp$ $\top$ $\perp$ $\perp$

* - Definition and images from Wikipedia, https://en.wikipedia.org/wiki/Bijection

$\{\text{enhanchment}\} \coprod \{\text{innovation}, \text{optimization}, \text{performance}, \ldots\}$

- Thanks to their relationship with entropy, set combinations can pose a significant challenge to deployment
- Expansion and reduction of entropy require compute resources
- The application of these compute resources is defined by code

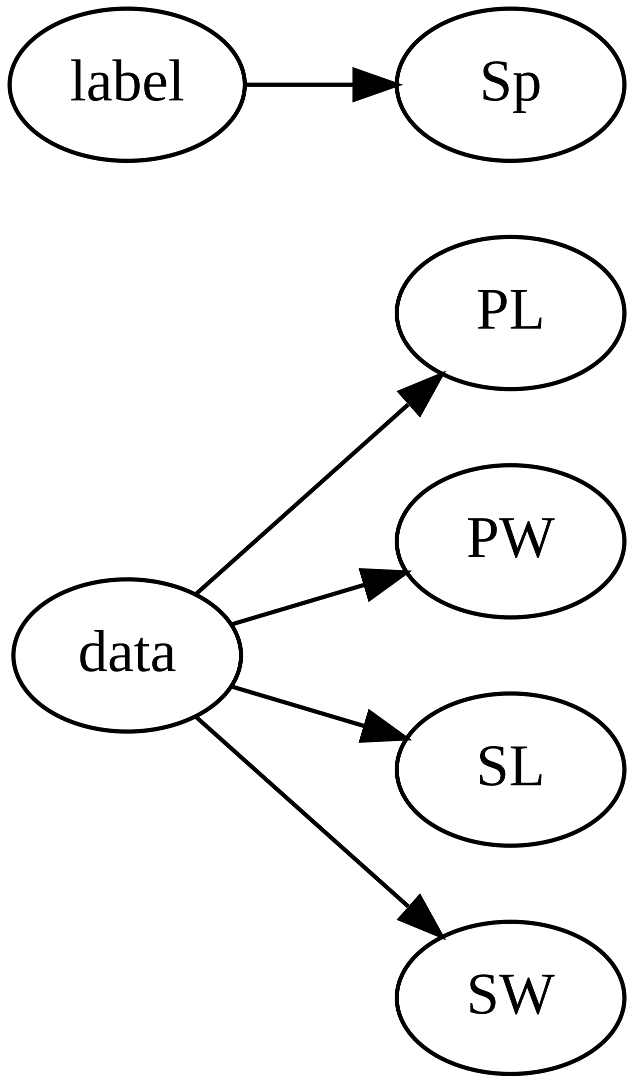
ergo,

1. Isomorphisms will not "free" us from performance issues, or necessarily increase performance

2. They are a relation defined by *theory*, so the difficulties of translating theory to practice apply

3. Working to realize stricter forms of relations across a system has a good deal of longitudinal value

Fischer's Iris Data Set*



$$data = \begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3.0 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5.0 & 3.6 & 1.4 & 0.2 \\ 5.4 & 3.9 & 1.7 & 0.4 \\ 4.6 & 3.4 & 1.4 & 0.3 \\ \dots & \dots & \dots & \dots \end{bmatrix} labels = \begin{bmatrix} setosa \\ setosa \\ setosa \\ setosa \\ setosa \\ setosa \\ setosa \\ \dots \end{bmatrix}$$

* - UCI Machine Learning Repository, 1988.

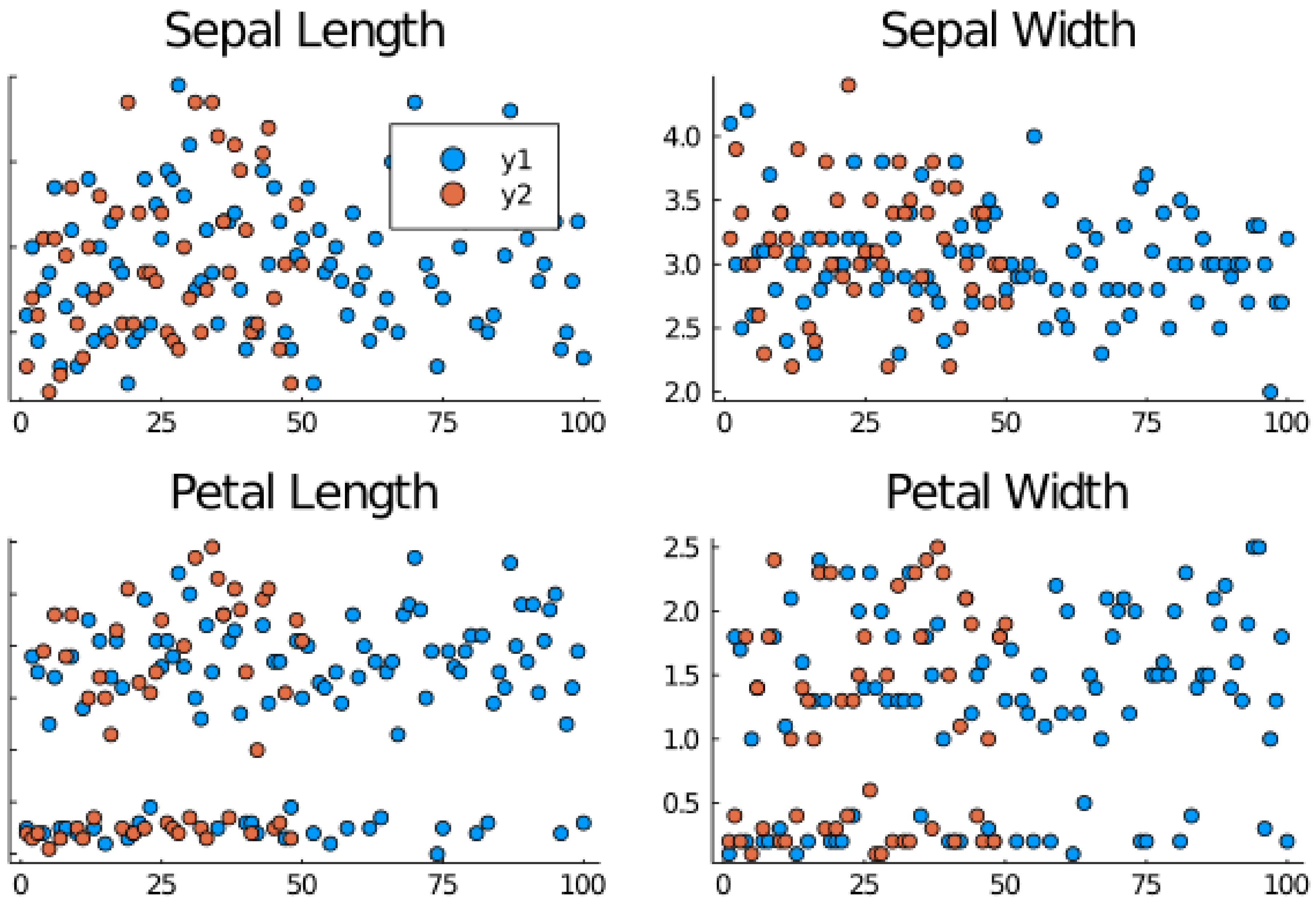
Relational View (6NF)

- Sixth Normal Form (6NF) was introduced by C.J. Date in the 1990s
- Excellent form for temporal data
- "[I]ntended to decompose relation variables to irreducible components."*

i	petal_length	i	petal_width	i	sepal_length	i	sepal_width	i	species
1	5.1	*1*	3.5	*1*	1.4	*1*	0.2	*1*	setosa
2	4.9	*2*	3.0	*2*	1.4	*2*	0.2	*2*	setosa
3	4.7	*3*	3.2	*3*	1.3	*3*	0.2	*3*	setosa
4	4.6	*4*	3.1	*4*	1.5	*4*	0.2	*4*	setosa
5	5.0	*5*	3.6	*5*	1.4	*5*	0.2	*5*	setosa
6	5.4	*6*	3.9	*6*	1.7	*6*	0.4	*6*	setosa
7	4.6	*7*	3.4	*7*	1.4	*7*	0.3	*7*	setosa
...	...	...	...	...	...	...	...	...	...

* - https://en.wikipedia.org/wiki/Sixth_normal_form

k-Nearest Neighbors (kNN) Classification



Naive kNN

$$
\begin{aligned}
&\forall(x \in X_{test} : \\
&\quad top[1, \\
&\quad\quad count[\ell : \\
&\quad\quad\quad labels_{Xtrain} \\
&\quad\quad\quad \wedge \\
&\quad\quad\quad top[k, \forall(y \in X_{train} : \\
&\quad\quad\quad\quad dist[x, y] \\
&\quad\quad\quad)] \\
&\quad\quad] \\
&\quad] \\
&)
\end{aligned}
$$

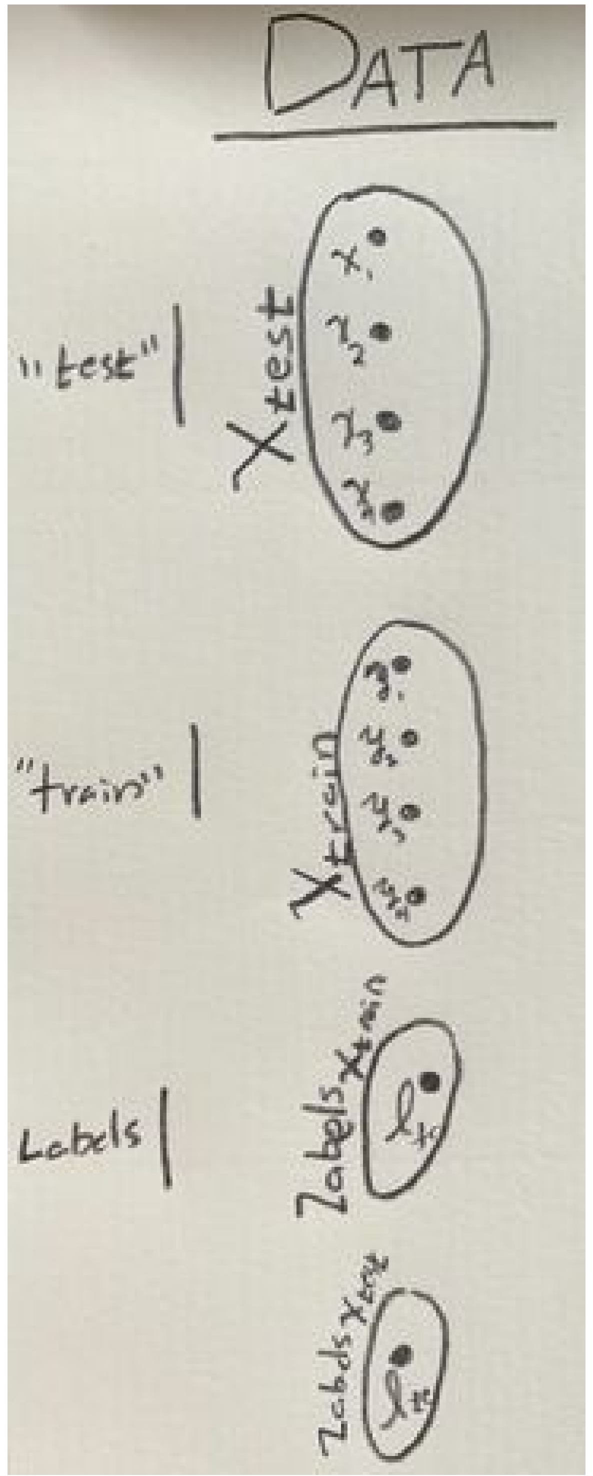
1. For each element of the "test" set, calculate a `dist`ance from each element in the "train" set
2. `sort` these via "nearest first" (lowest `dist` value) then take the `top` k tuples
3. `join` the top k tuples with their $labels$
4. `count` the number of each label in the set just formed
5. `sort` these via "most frequent" then take the first (`top[1,...]`) tuple in the relation

Now you know why naive kNN scales at $O(dn^2)$

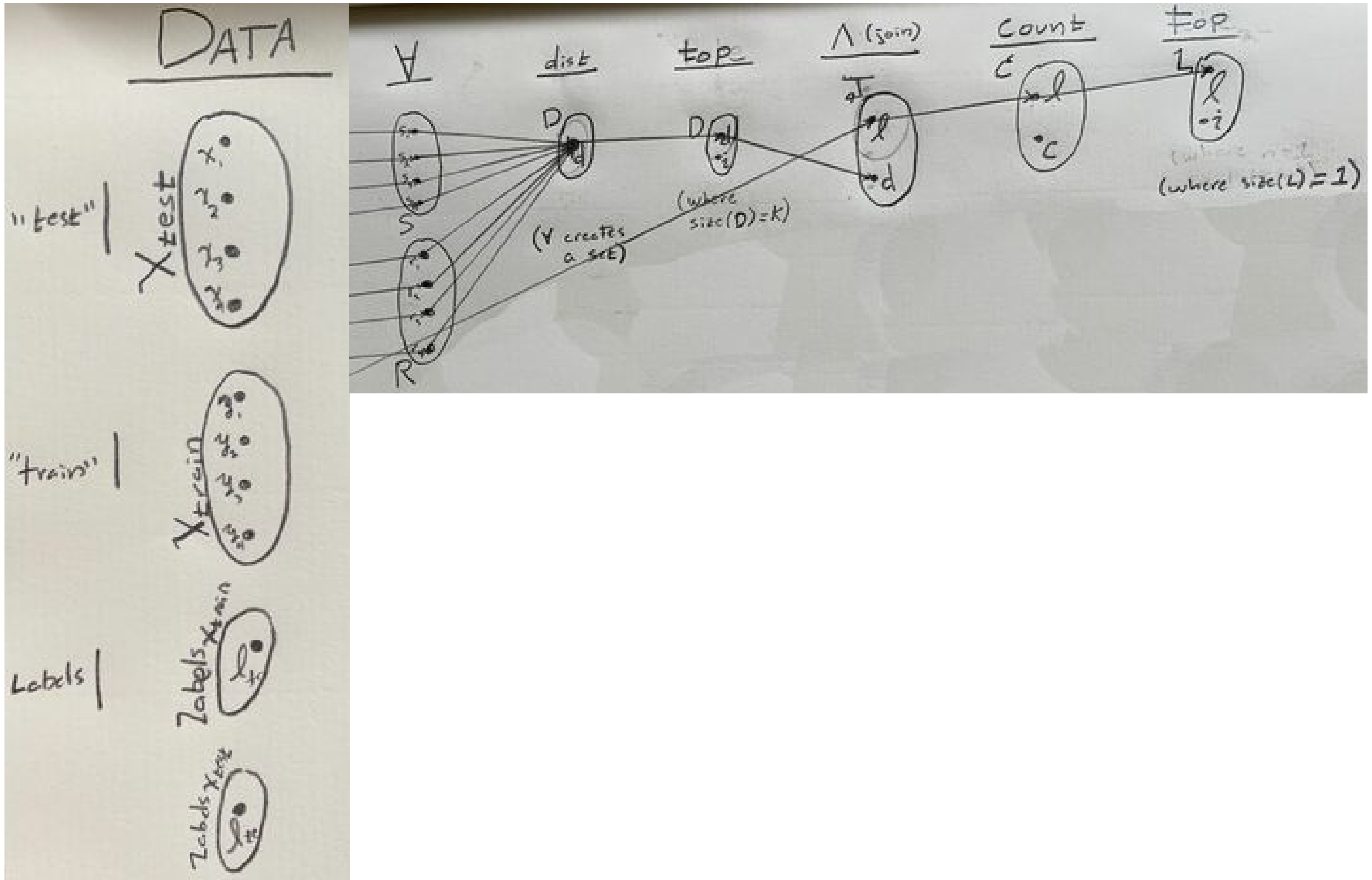
In our case that's $4 \cdot 150^2 = 4 \cdot 22500 = 90000$

for *150* tuples of airity *4*

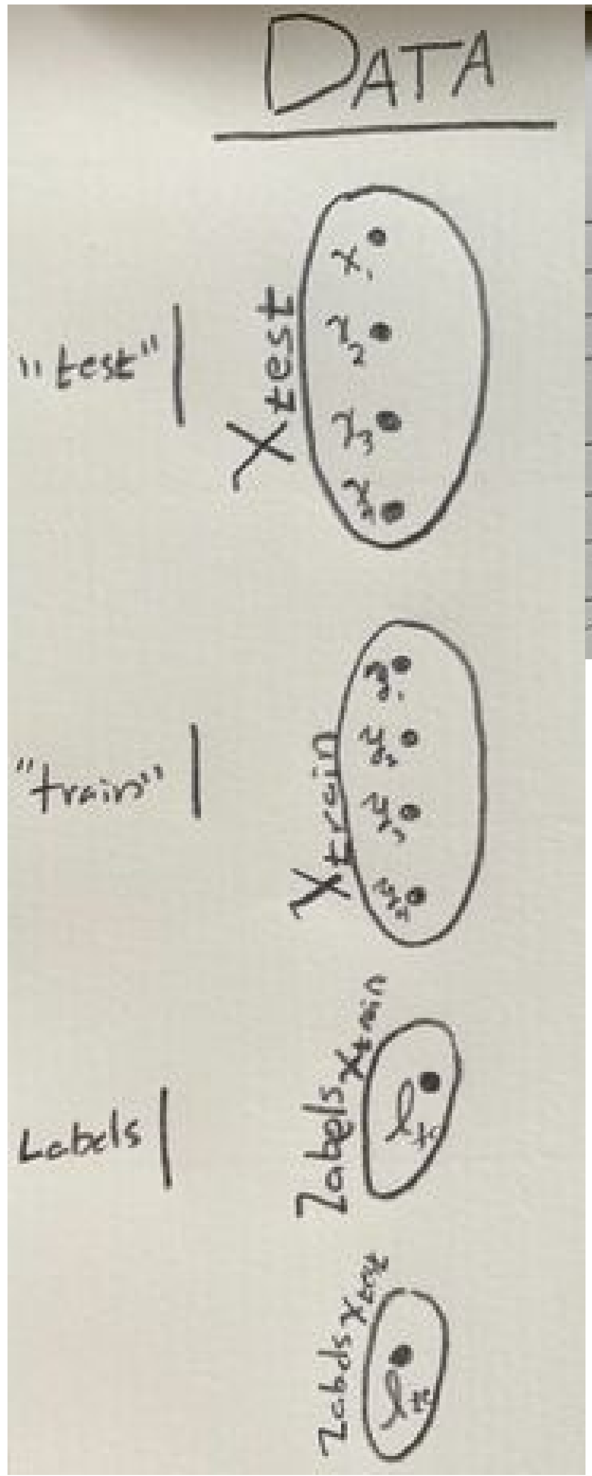
$$data_{input} \rightarrow kNN() \rightarrow labels_P \rightarrow viz()$$

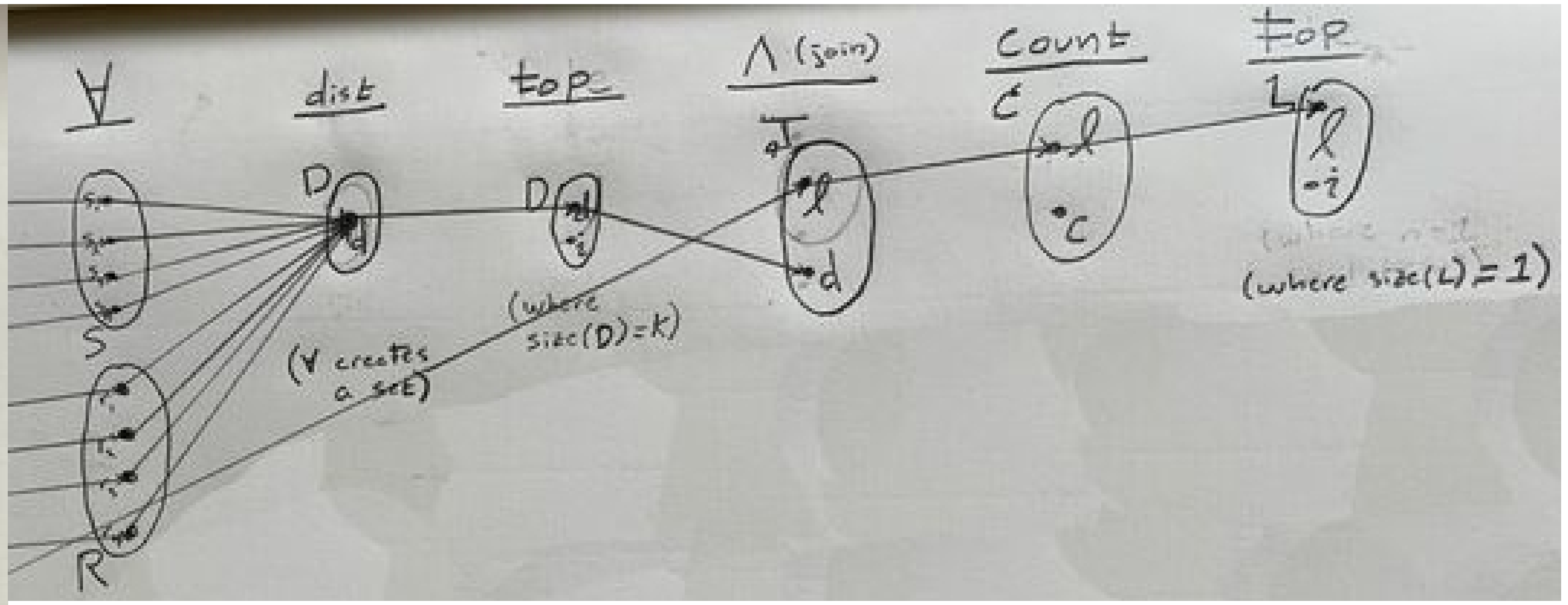


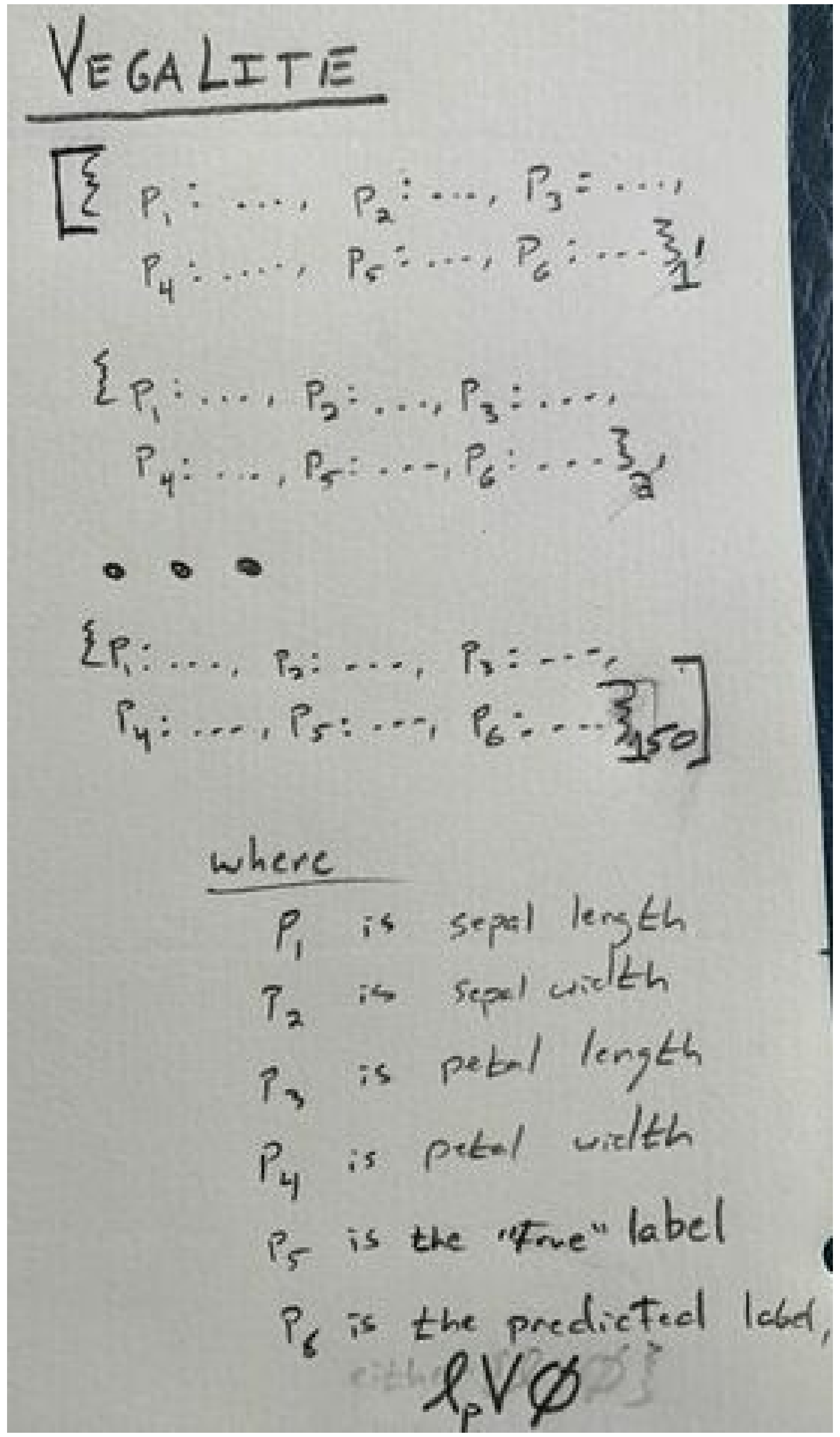
$$data_{input} \rightarrow kNN() \rightarrow labels_P \rightarrow viz()$$



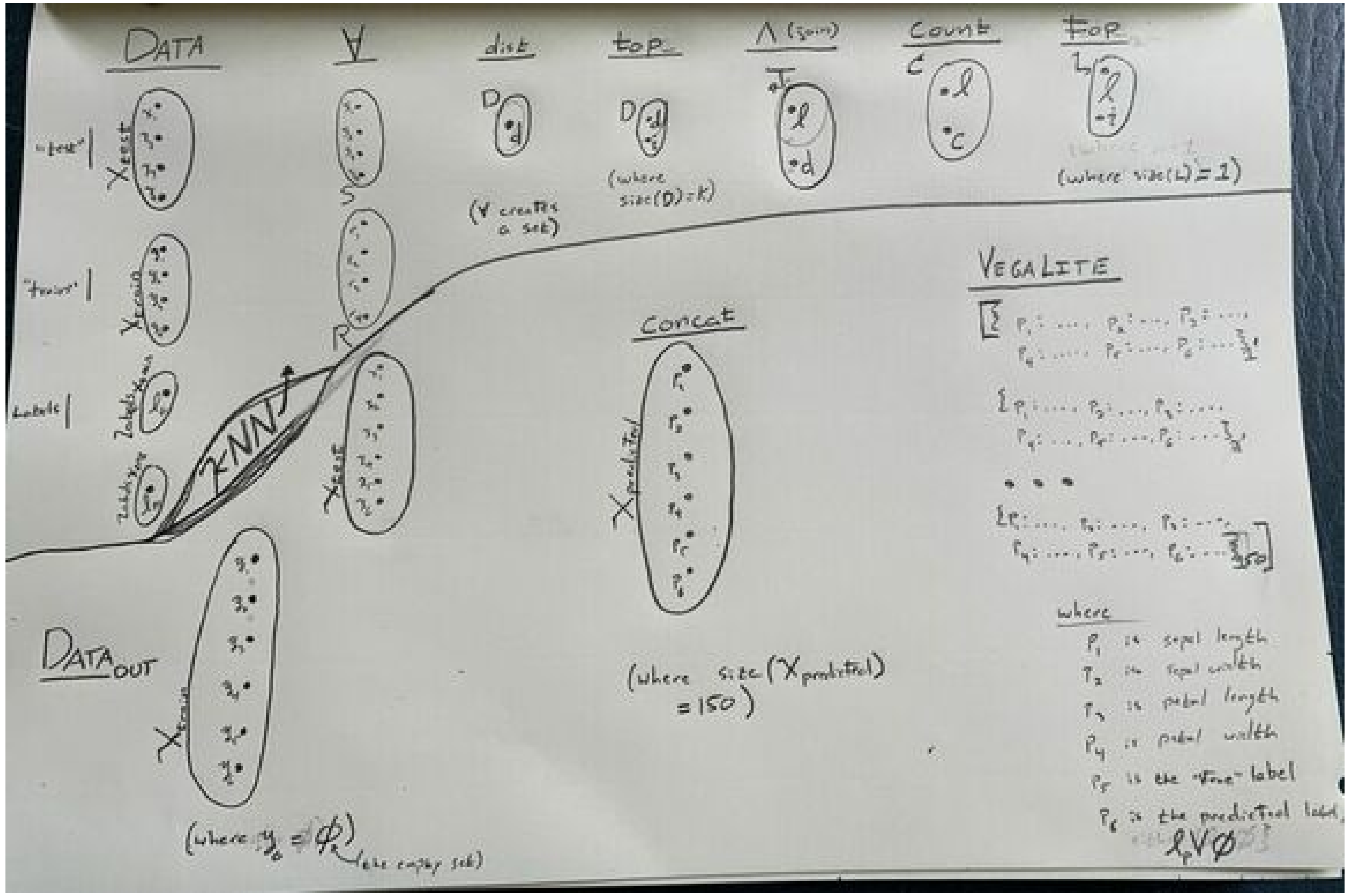
$$data_{input} \rightarrow kNN() \rightarrow labels_P \rightarrow viz()$$



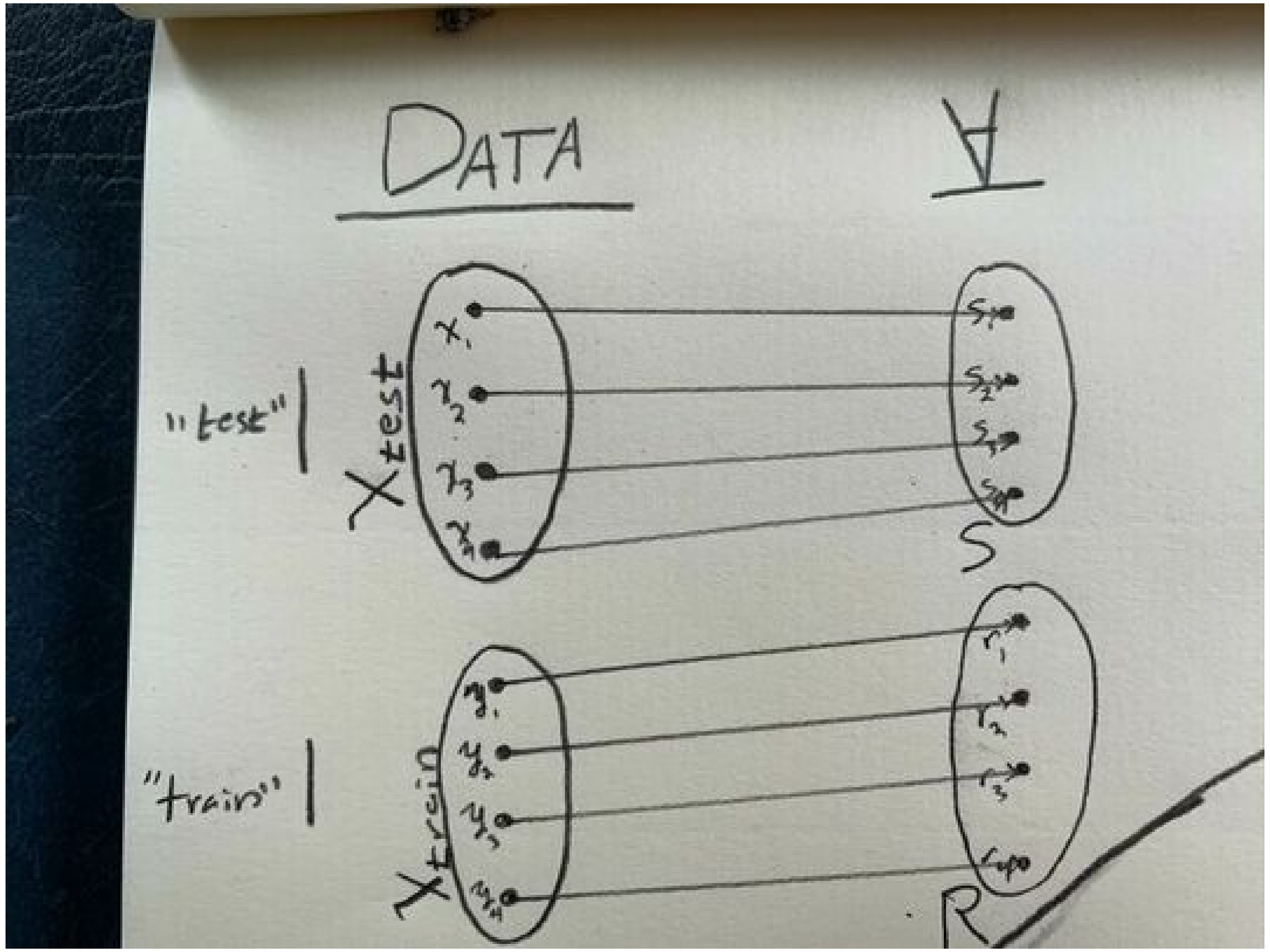




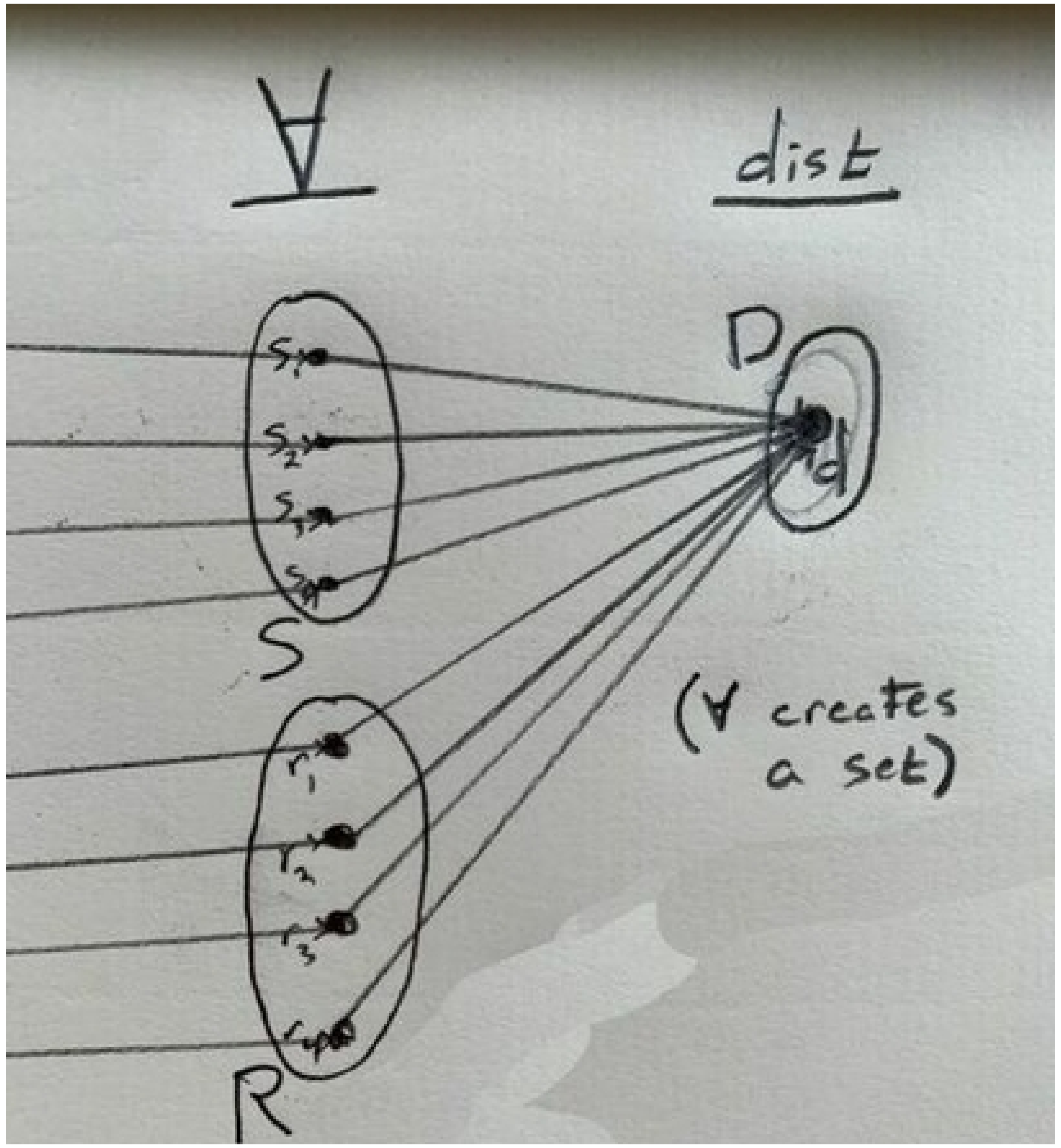
End-to-End Operation



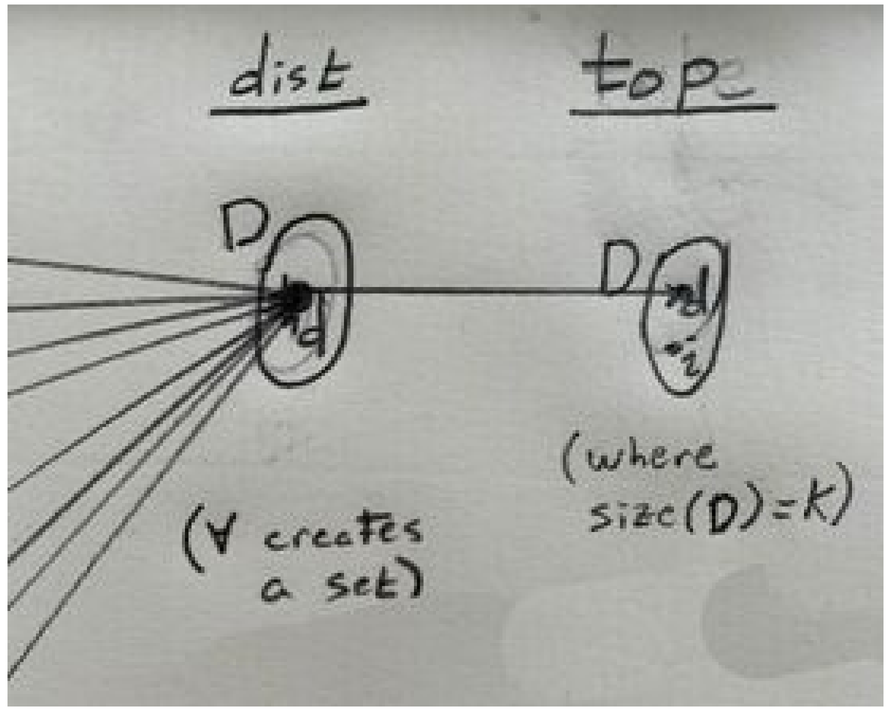
$$data \rightarrow \forall$$



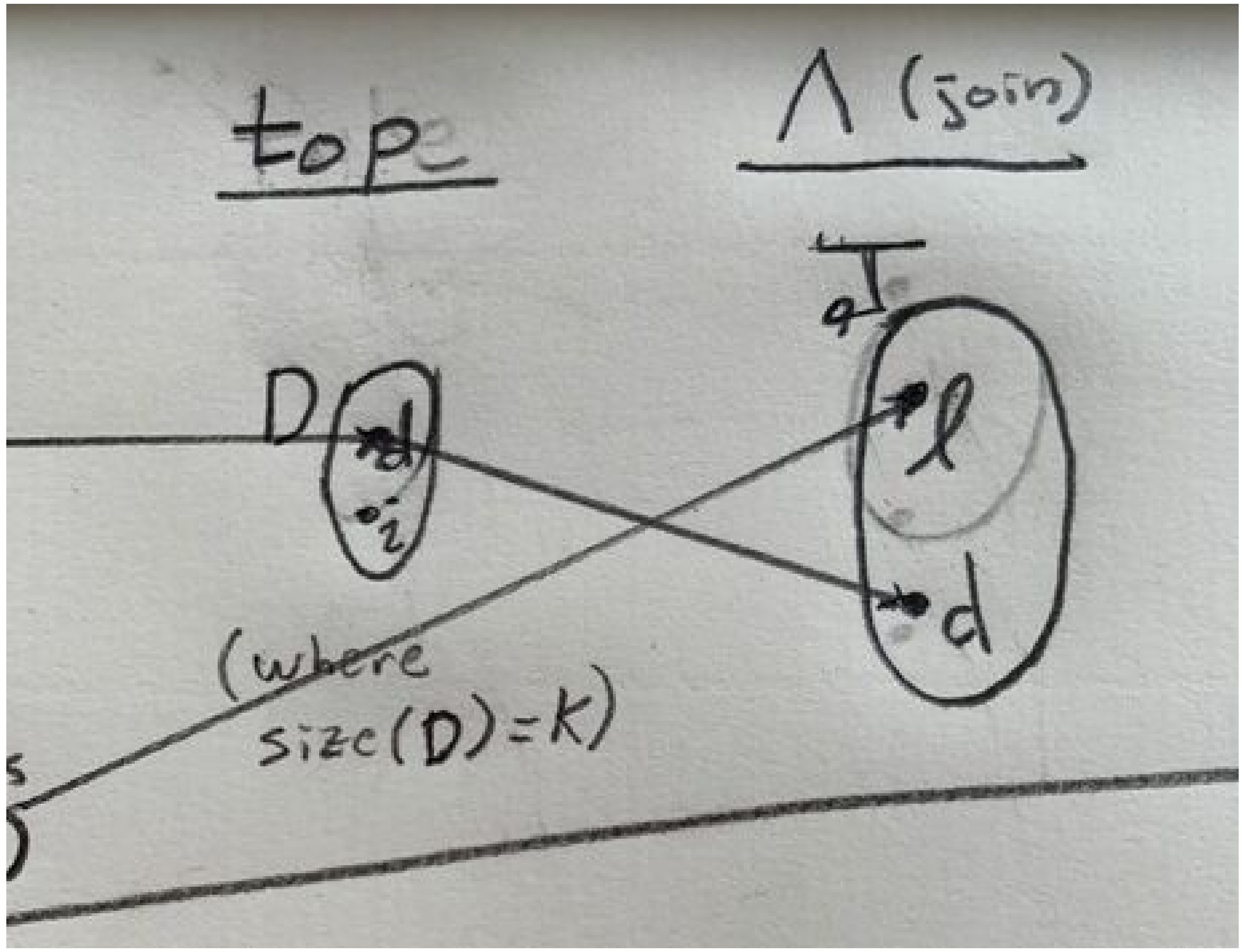
$\forall \rightarrow dist$



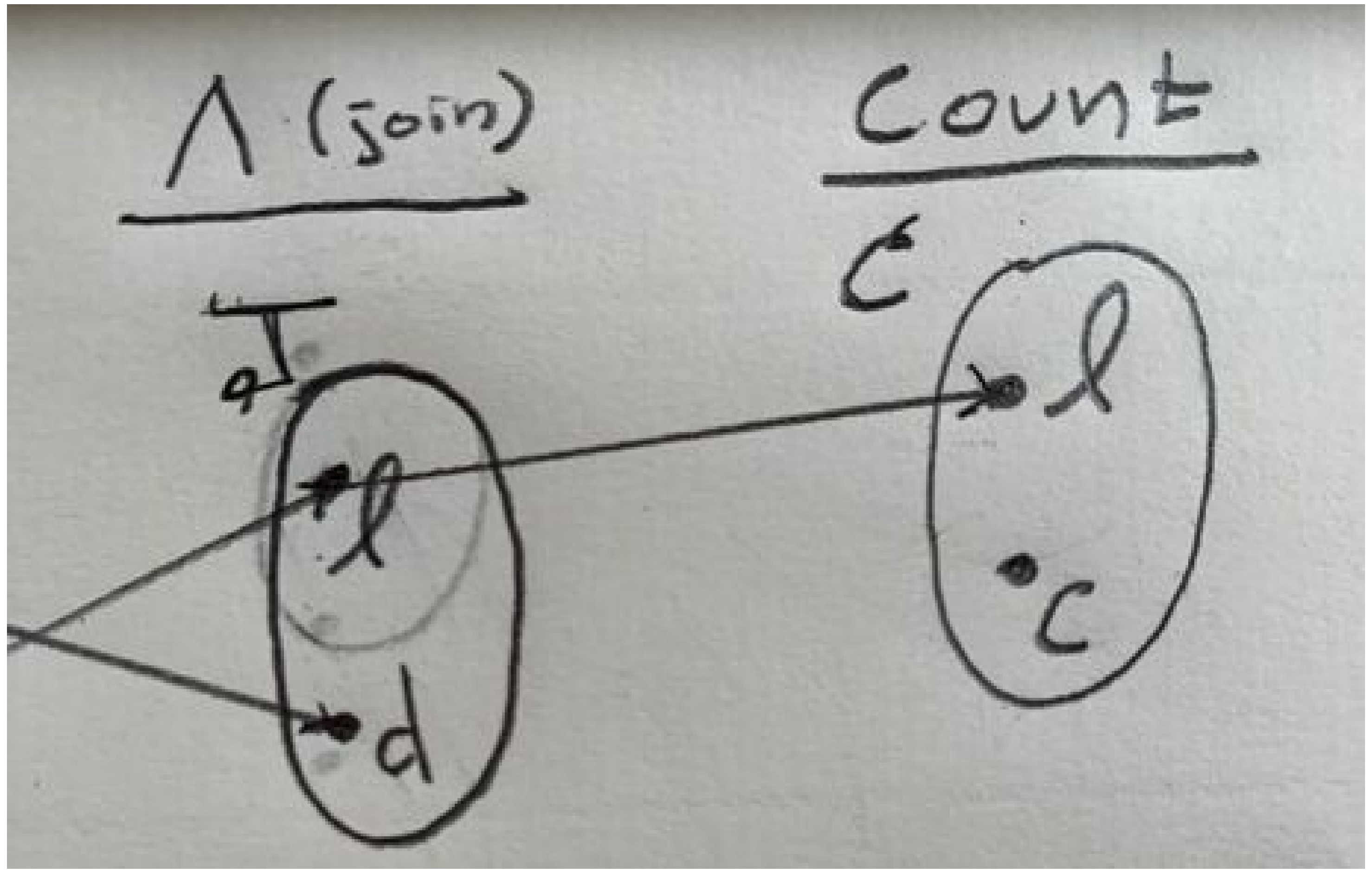
$$dist \rightarrow top$$



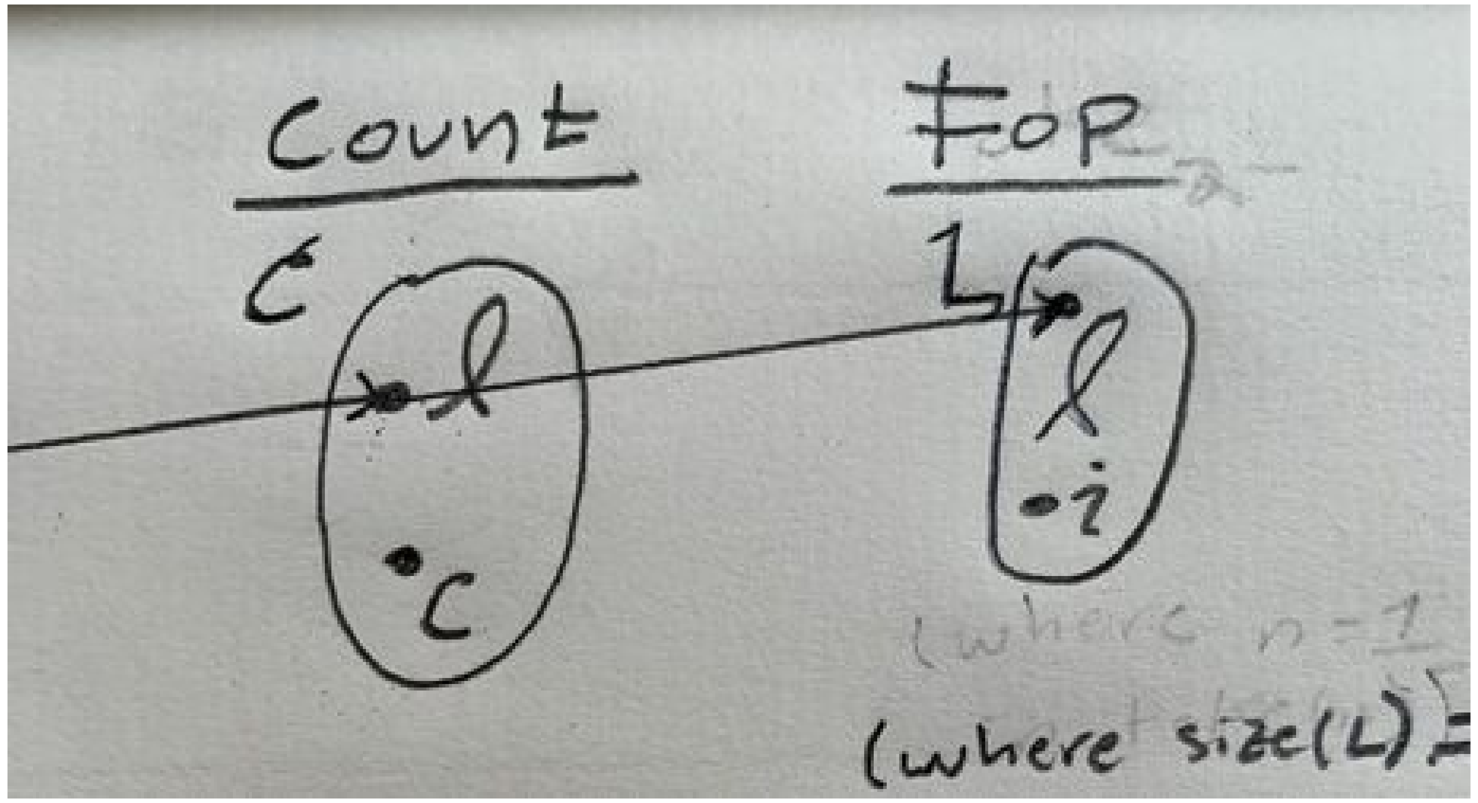
$top \to \wedge$



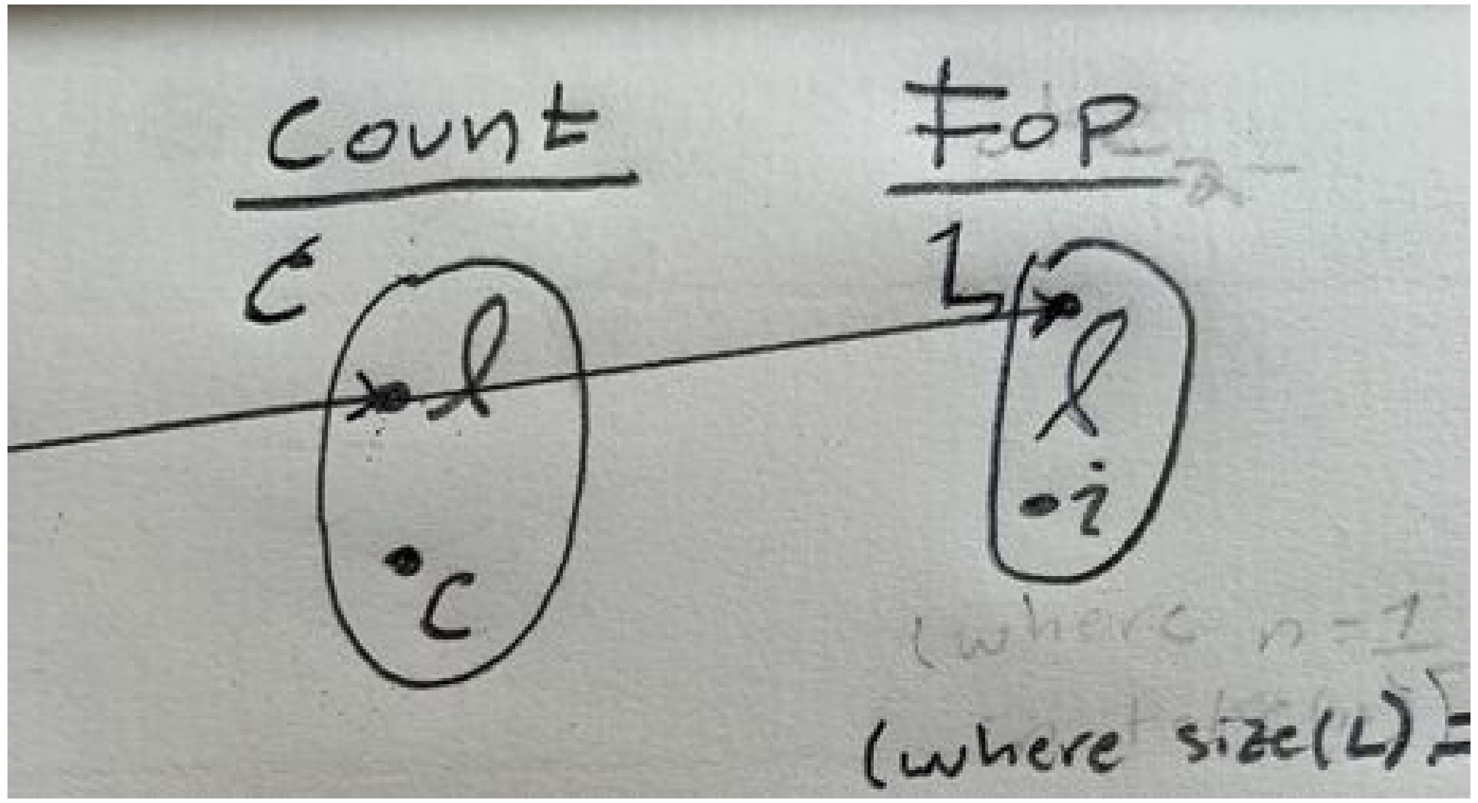
$\wedge \to count$



$$count \to top$$



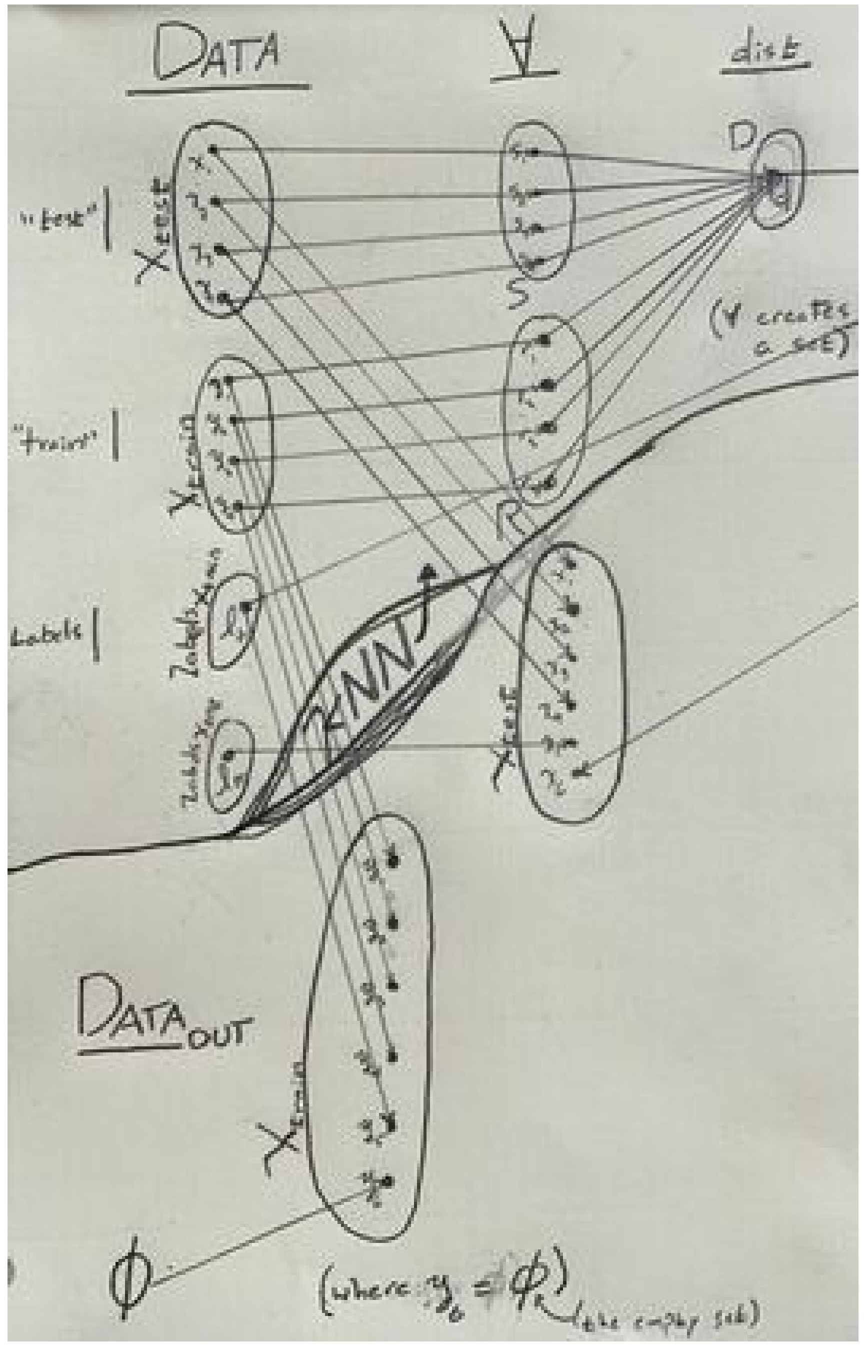
$count \rightarrow top$



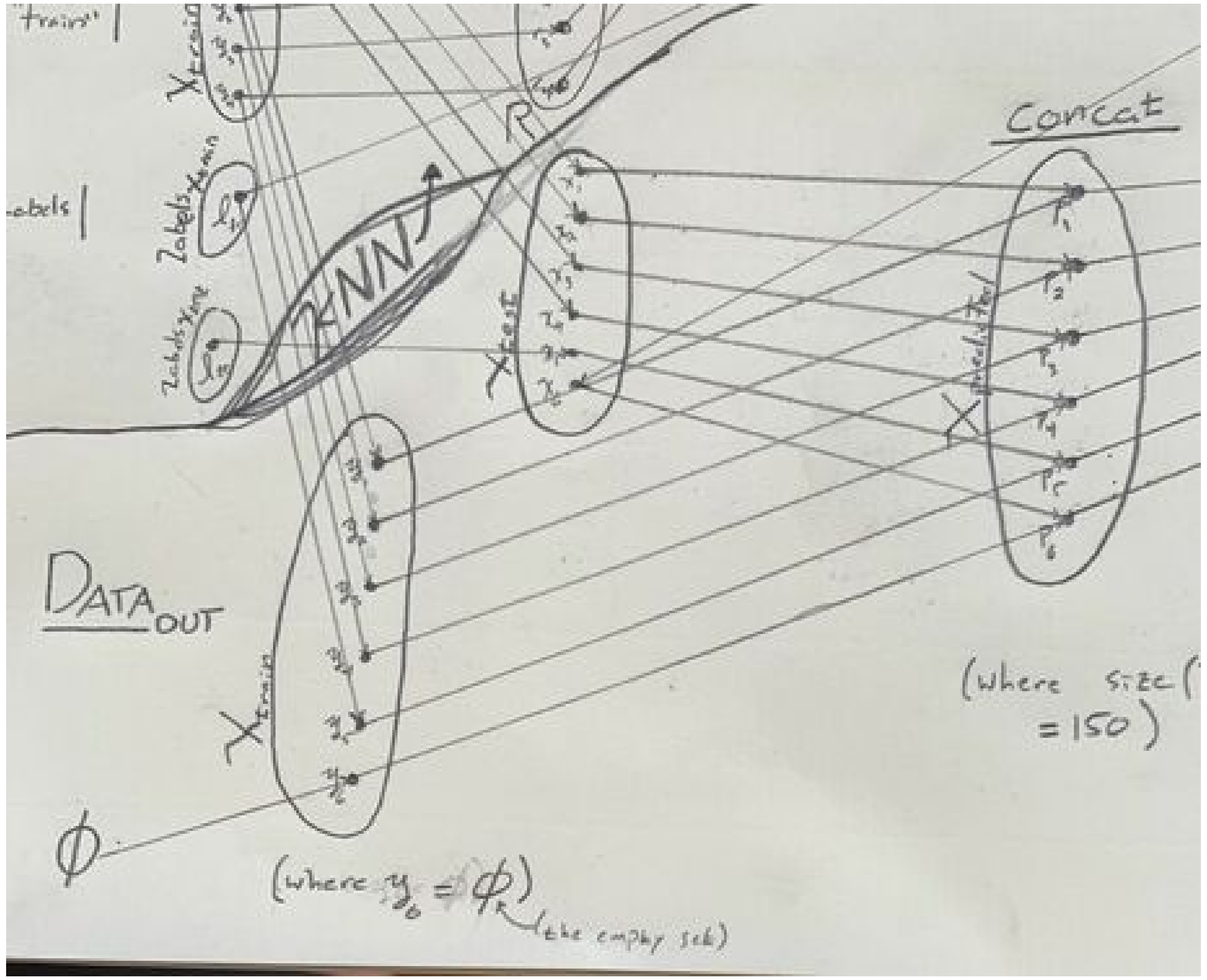

```
// out =>
```

$$labels_P = [\,\ell_1;\ \ell_2;\ \ell_3;\ \ell_4;\ \ell_5;\ \ell_6;\ \ell_7;\ \ldots;\ \ell_n\ ]$$

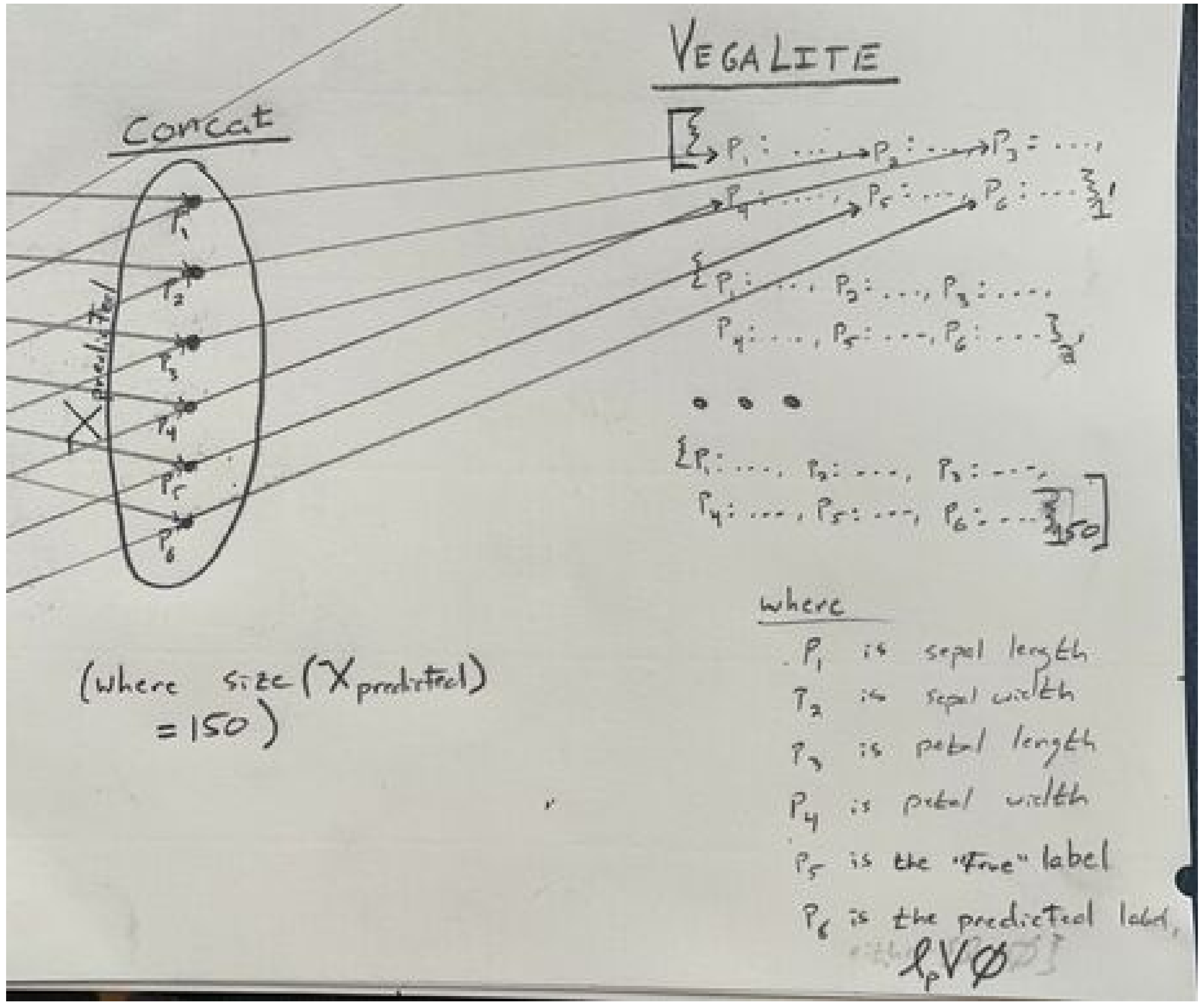
$$data \to data_{out}$$



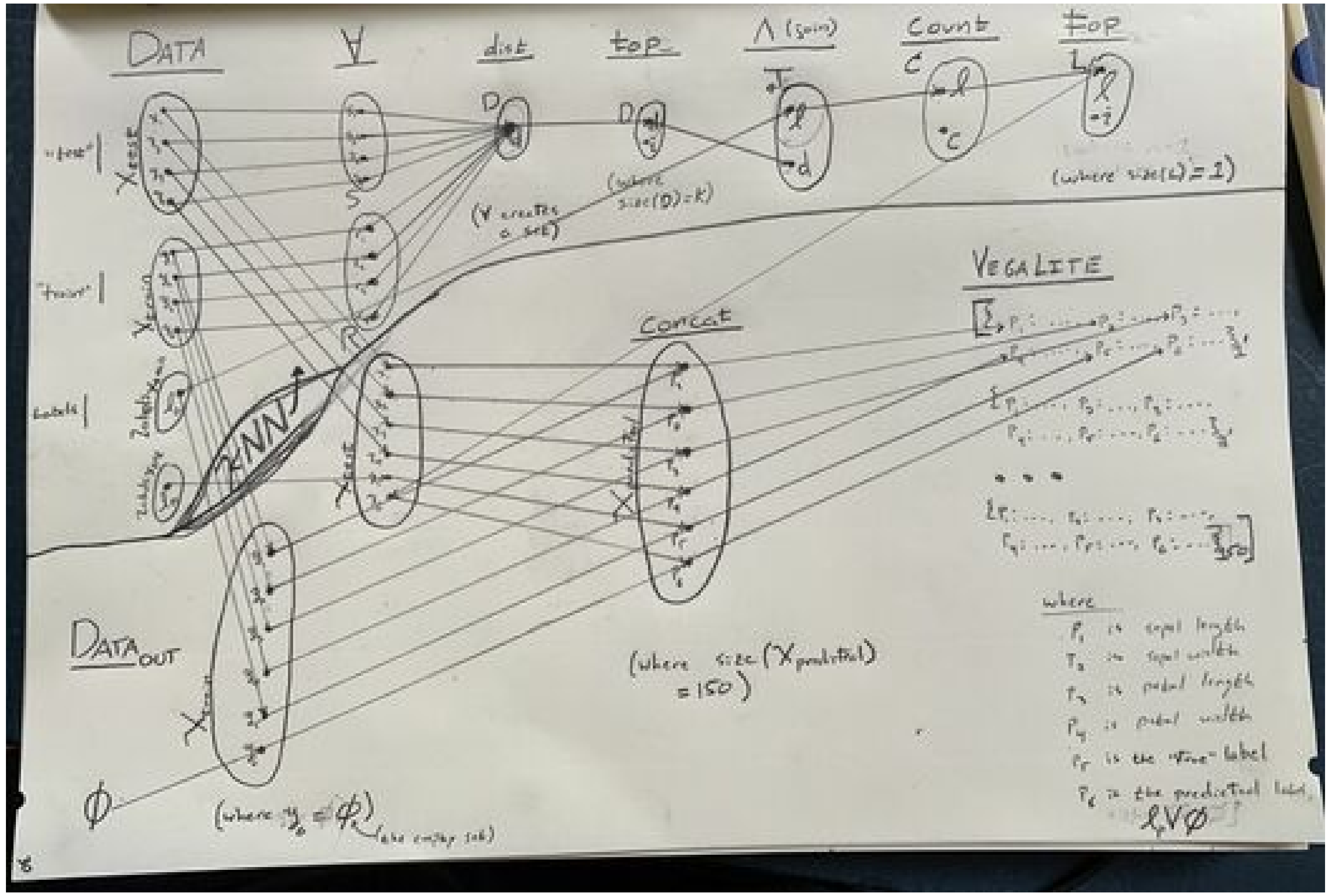
$$data_{out} \rightarrow concat$$



$$concat \rightarrow VegaLite$$



Finished Operation



Enhancements

1. Accountability of operations
2. Easier identification of potential fail points
3. Identification of areas that generate excessive facts
4. Resource allocation and planning
5. Stable interface for data vizualization API
    - Vega-Lite, in our case

References

1988, Dua, Dheeru, and Graff, Casey **(view online)**
UCI Machine Learning Repository

Appendix

Euclid distance for tuples of airity n

$$
\begin{aligned}
&\exists(r_1, \ldots, r_n \in R \wedge s_1, \ldots, s_n \in S:\\
&\quad sqrt[\\
&\quad\quad sum[\\
&\quad\quad\quad squared[\\
&\quad\quad\quad\quad (s_1 - r_1; \ldots; s_n - s_n)\\
&\quad\quad\quad]\\
&\quad\quad]\\
&\quad]\\
&)
\end{aligned}
$$