* Working with categorical data :

Categorical data has a specific set of possible values.

→ The different species of animals in a national park

→ If an email is spam or not

→ The names of streets in a city

* Encoding : It means converting categorical or other data to numerical vectors that a model can train on. The model can only train on floating point values; models can't train on strings such as "dog" (or) "maple".

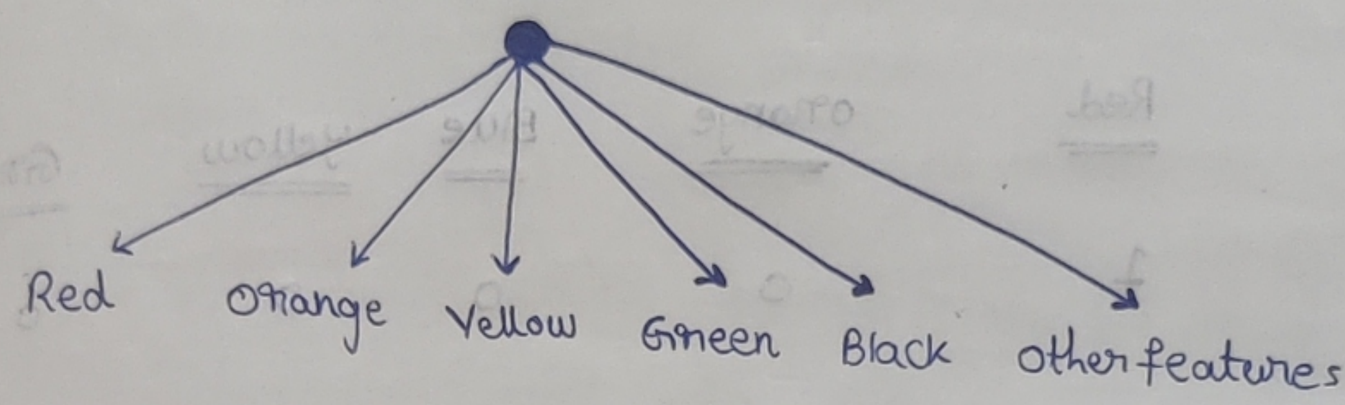
* Vocabulary and one-hot encoding :

"Dimension" is a synonym for the number of elements in a "feature vector."

Feature_Name	# of categories	Sample categories
snowed_today	2	True, False
skill_level	3	Beginner, Practioner, Expert
season	4	Winter, Spring, Autumn, Summer
day-of-week	7	Monday, Wednesday, Thursday
planet	8	Mercury, Venus, Earth

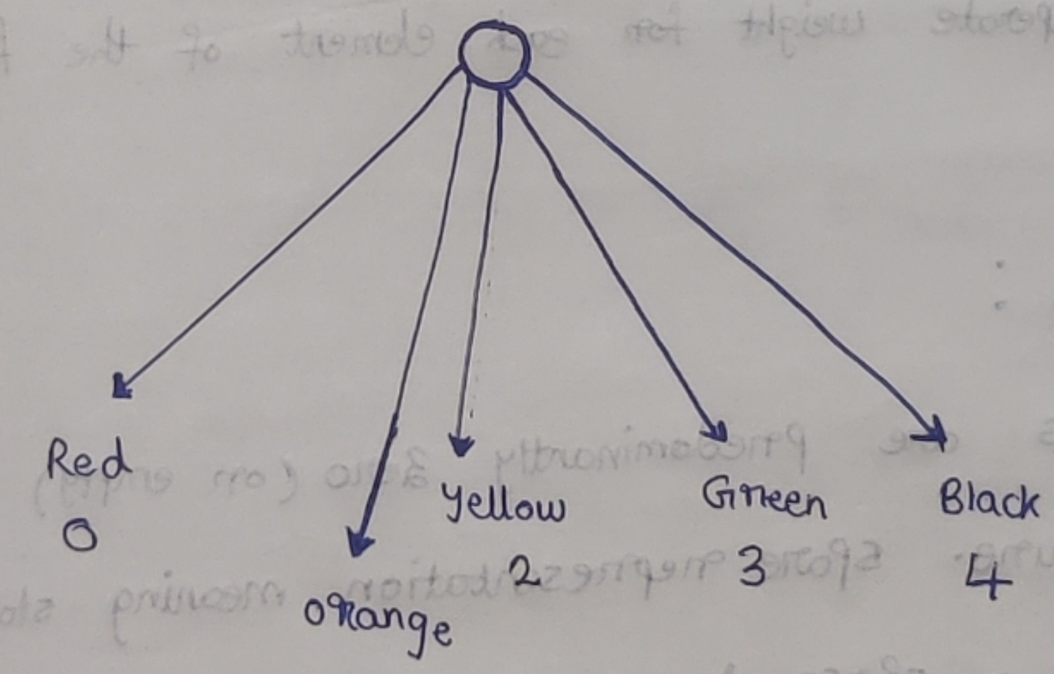
When a categorical feature has a low number of features in a category, we can use "Vocabulary encoding". With this the model treats each possible categorical value as seperate feature. During training it learns different weights of each category.

suppose we want to create a model to Prdict cars price based on colour of the car. The car-colour is a low-dimensional categorical feature.



* Index Numbers :

As we know ML models can only manipulate floating-point numbers. Therefore, we must convert each string to unique index number.



we'll need to process the data further to represent in ways that help the model learn meaningful relationships between the values. If the categorical feature data is left as indexed integers and loaded into a model, the model would treat the index value as continuous floating point numbers.

* one-hot encoding :

The next step is building a vocabulary is to convert each index number to it's one hot encoding. In this :

→ Each category is represented by a vector of N elements, where N is the number of categories. If car_colour has 8 possible categories, the one-hot encoding will have 8 elements.

→ Exactly one of the elements in a one-hot vector has the value of 1.0 all the remaining elements have value 0.0.

Feature	Red	orange	Blue	yellow	Green
Red	1	0	0	0	0
orange	0	1	0	0	0
Blue	0	0	1	0	0
Yellow	0	0	0	1	0
Green	0	0	0	0	1

the model learns a seperate weight for each element of the feature vector.

* Sparse representation :

A feature whose values are predominantly zero (or empty) is termed a sparse feature. Spare representation meaning storing the position of the 1.0 in a sparse vector. For example, the one-hot vector for Blue is : [0, 0, 1, 0, 0]

As. 1 is in position 2, the sparse representation of preceding one -hot vector is : 2

sparse representation consume far less memory than one-hot vector but the model must train on one-hot vector not sparse representation.

* Outliers in categorical data :

Like numerical data, categorical data also contains outliers. for ex, in the car_colour case apart from normal colours there are some rare colours like "Mauve" (or) "Avocado". Rather than giving them a each category we can rump them into a single "catch-all" category called out-of-vocabulary (OOV)

"

When the number of categories is High, one hot encoding is a bad choice, Embeddings are a much better choice".

Benefits of embeddings :

→ the model trains faster

→ The built model typically infers Predictions more quickly.

i.e, the model has low latency.

" "

Hashing is a less common way to reduce the number of dimensio

Human raters : Data often labeled manually by humans is referred as "golden labels" and is considered more desirable than ML data. But humans also do mistakes so, Any two human beings may label the same example differently.

→ The difference between human raters' decisions is called inter-rater agreement.

The following are ways to measure inter-rater agreement :

→ Cohen's kappa and variants
→ Intra-class correlation (ICC)
→ Krippendorf's alpha

Machine raters : Machine-labeled data, where categories are automatically determined by one (or) more classification models is often referred to as silver labels.

High dimensionality : Categorical data tends to produce high-dimensional feature vectors, i.e, feature vectors having large number of elements. For natural language data, the main method of reducing dimensionality is to convert feature vectors to embedding vectors.

* Feature crosses :

Feature crosses are "created by crossing (taking cartesian Product)" of two (or) more categorical or bucketed features of the dataset.

→ Feature crosses allow linear models to handle nonlinearities.
→ Feature crosses also encode interactions between features.

for example, consider a leaf dataset with categorical features :

→ edges : smooth, toothed, lobed
→ arrangement : opposite, alternate

so let us take a one-hot representation of "smooth-edged and opposite arrangement leaf" so it will be { (1,0,0) , (1,0) }

The feature cross/cartesian Product will be :

{ smooth_opposite , smooth_alternate , tooth_opposite , tooth_alternate , lobed_opposite , lobed_alternate }

→ smooth_opposite = edges[0] * arrangement[0]
→ smooth_alternate = edges[0] * arrangement[1]
→ toothed_opposite = edges[1] * arrangement[0]
→ toothed_alternate = edges[1] * arrangement[1]
→ lobed_alternate = edges[2] * arrangement[1]
→ lobed_opposite = edges[2] * arrangement[0]

for ex, if a leaf has a lobe edged alternative arrangement the cross feature will be {0,0,0,0,0,1}