Implicit Feature Engineering

- is when the ML model itself automatically transforms or expands features internally, without you explicitly creating them.
- Ex: say you have 2 features: x_1, x_2 → P-7
  in explicit feature engineering you might add for ex: x_1^2, x_2^2, x_1x_2
  in implicit feature eng the model does this for you internally (no-code)

- Where? is it used?

Model	How it does implicit Feature Eng
Kernal SVMs P-22	use the kernal trick to map inputs into high dim space implicitly (eg RBF adds infinite non-linear functions)
Neural Networks	hidden layers learn non linear transformation of features automatically (Deep Learning)
Decision trees / random forests	learn interaction effects (splits on combination of features) without you encoding them
Polynomial features + linear Model	you can make it explicit (Polynomial features) or implicit via the model kernal

Important NOTE on LLM embedding and decoding Mappings

- Word to vector (embedding) Mapping
  - During input each token (often subword not whole word every time) is converted into a fixed embedding vector the model has learned
  - This is exact: the token "cat" always maps to the same embedding vector
- Vector to token (decoding) mapping
  - During output the LLM produces a output vector influenced by the entire vocabulary and the input
  - This vector is not exact (Probalistic) its the models ouput for what it wants to generate
  - The output token is chosen by finding closest match in vocabulary by a softmax over Logits not a strict nearest neighbour in vector space

- NOTE: its not litrally the closest vector in a space sphere in most implementations its more Like a dot product (cos similarity) with each embedding to compute Logits then sampling (pick 111 random token according to probabilities) or argmax (pick token with highest probability)