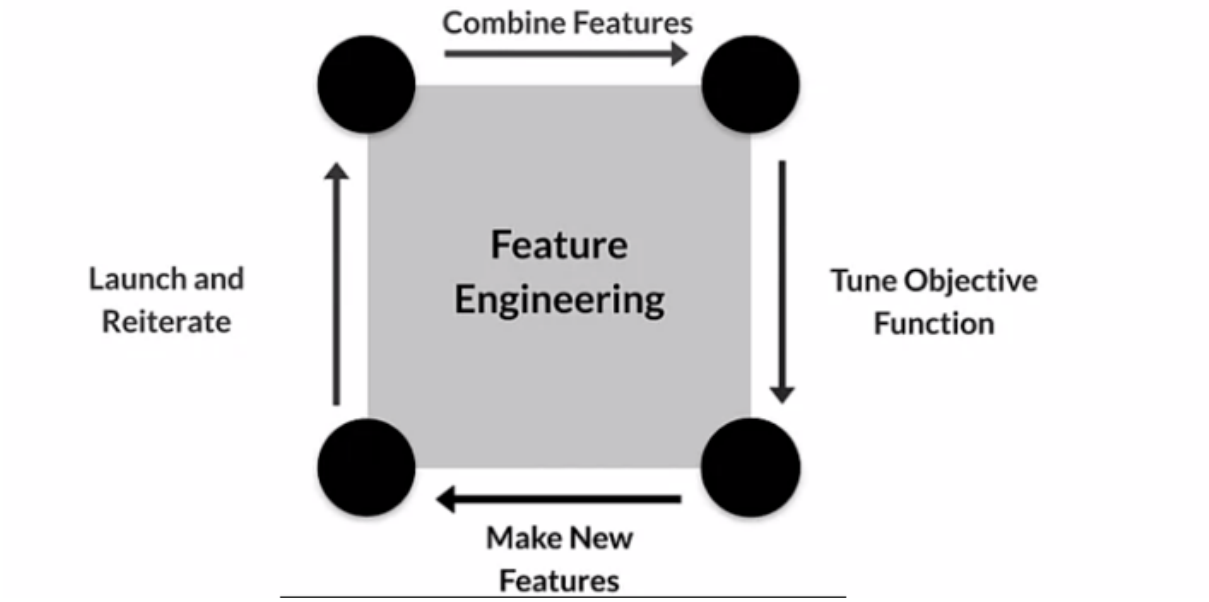
Week 2 - Preprocessing, Feature Engineering, Feature Transformation at Scale, Feature Selection.



When you perform feature engineering in a training environment, you need to do the same preprocessing steps in serving.

- Data preprocessing: transforms raw data into a clean and training ready dataset
- Feature engineering maps:
    - Raw data into feature vectors
    - Integer values to floating point values
    - Normalizes numerical values
    - Strings and categorical values to vectors of numeric values
    - Data from one space into a different space

Preprocessing Operations:

Empirical Knowledge of data:

Text - stemming, lemmatization, TF-IDF, n-grams, embedding lookup

Images - clipping, resizing, cropping, blur, Canny filters, Sobel filters, photometric distortions

Feature Engineering:

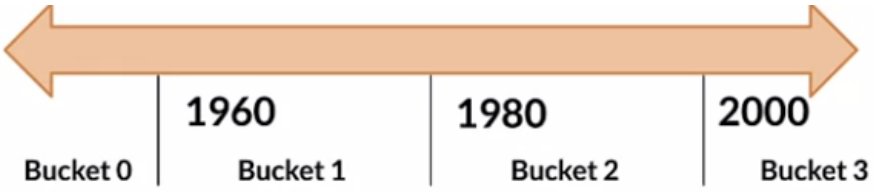
- Scaling:
  - Converts values from their natural range into a prescribed range
    - Grayscale image pixel intensity scale is [0, 255] usually rescaled to [-1,1]
- Benefits:
  - Helps neural nets converge faster
  - Do away with NAN errors during training
  - For each feature, the model learns the right weights
- Normalization: When data is not gaussian, it is a good starting point (rule of thumb)

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

- Standardization (z-score)
  - Z-score relates the number of standard deviations away from the mean

$$X_{\text{std}} = \frac{X - \mu}{\sigma} \qquad \text{(z-score)}$$

- Bucketizing/Binning



Date Range	Represented as...
< 1960	[1, 0, 0, 0]
>= 1960 but < 1980	[0, 1, 0, 0]
>= 1980 but < 2000	[0, 0, 1, 0]
>= 2000	[0, 0, 0, 1]

- Dimensionality reduction in embeddings:
  - PCA
  - t-SNE
  - UMAP
- Feature Crossing: Combines multiple features together into features.

Tools:

- Facets (Visualization)
- TensorFlow embedding projector
  - Intuitive exploration of high-dimensionality data

- Visualize and analyze
- Techniques
  - PCA
  - t-SNE
  - UMAP
  - Custom linear projections
- Ready to play

Preprocessing data at Scale:

Transformations	
Instance-level	**Full-pass**
Clipping	Minimax
Multiplying	Standard scaling
Expanding features	Bucketizing
etc.	etc.

Pre-processing training dataset:

Pros	Cons
Run-once	Transformations reproduced at serving
Compute on entire dataset	Slower iterations

Transforming within the model:

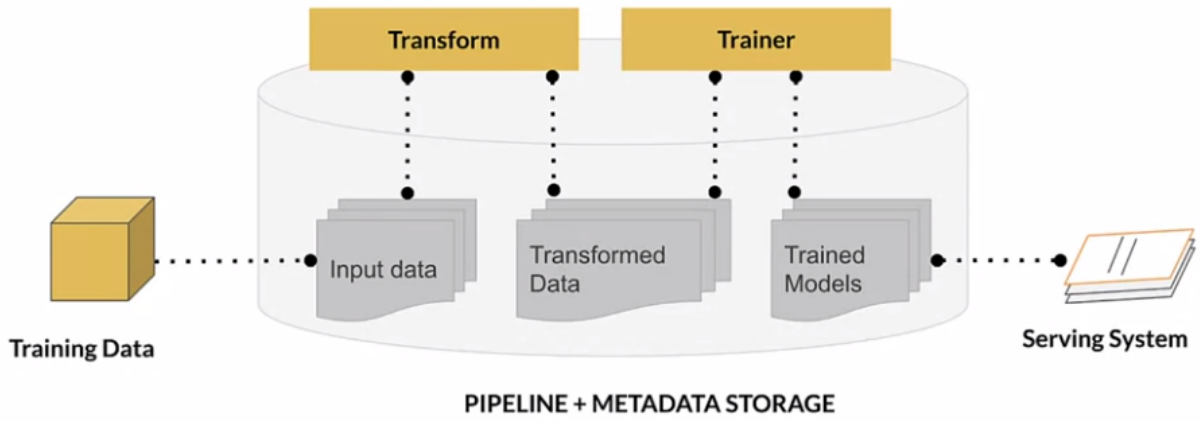
Pros	Cons
Easy iterations	Expensive transforms
Transformation guarantees	Long model latency
	Transformations per batch: skew

Transform per batch:

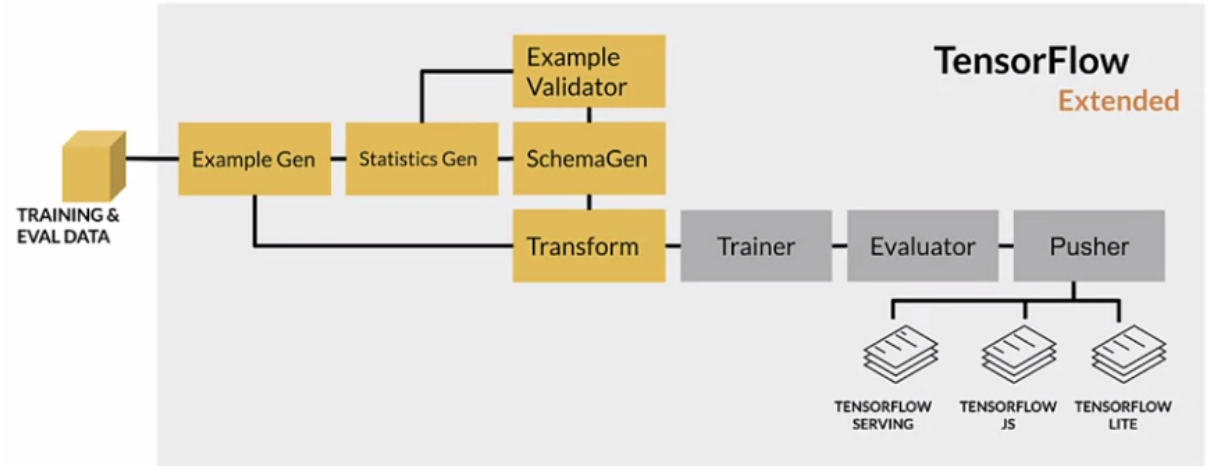
- Normalizing features by their average
- Access to single batch of data, not full dataset
- Ways to normalize per batch:
    - Normalize by average within a batch
    - Precompute average and reuse it during normalization
- Indirectly affect training efficiency
- Typically accelerators sit idle while CPUs transform
- Solution:
    - Prefetching transforms for better accelerator efficiency.

Tensorflow Transforms:

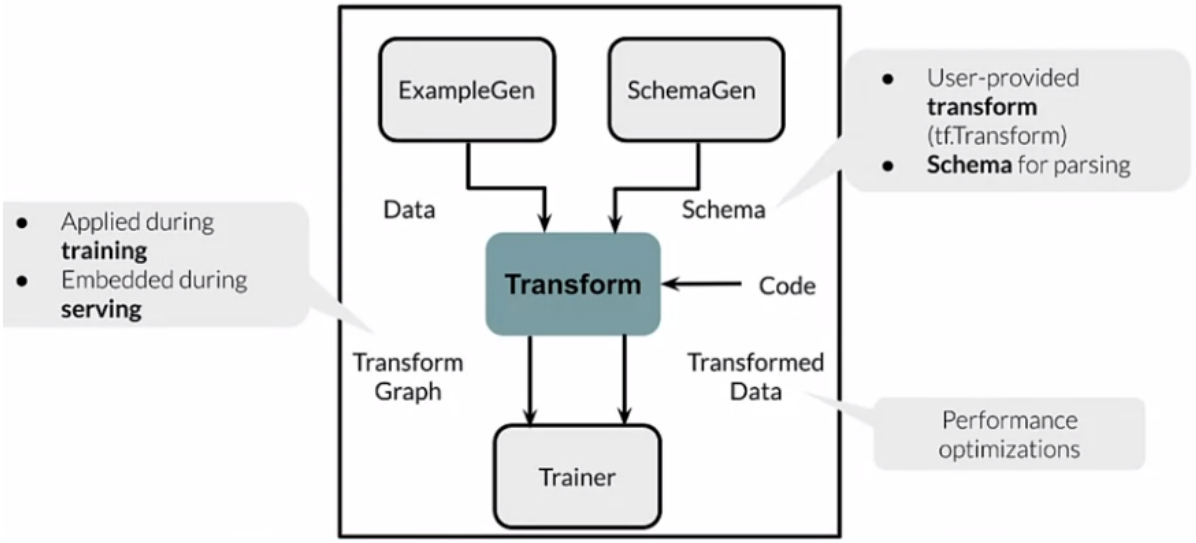
tf.Transform:



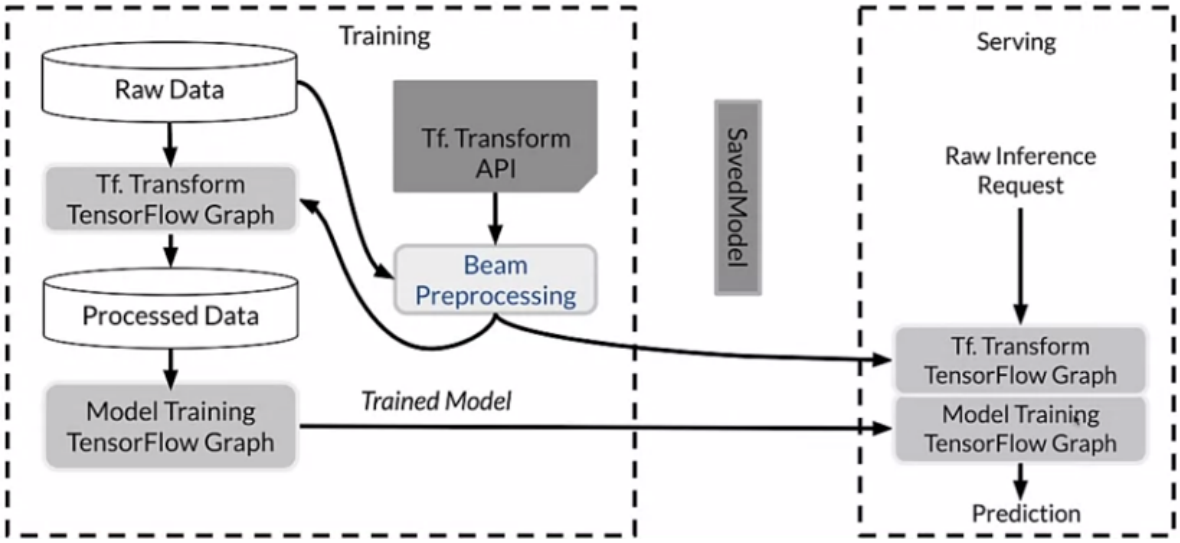
Inside TensorFlow Extended:



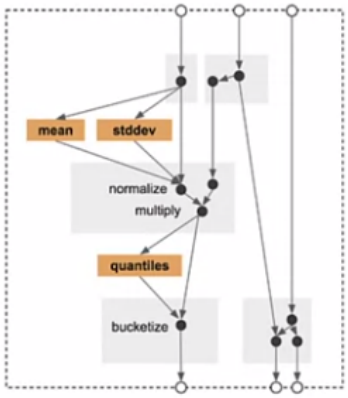
Layout:



Going Deeper:



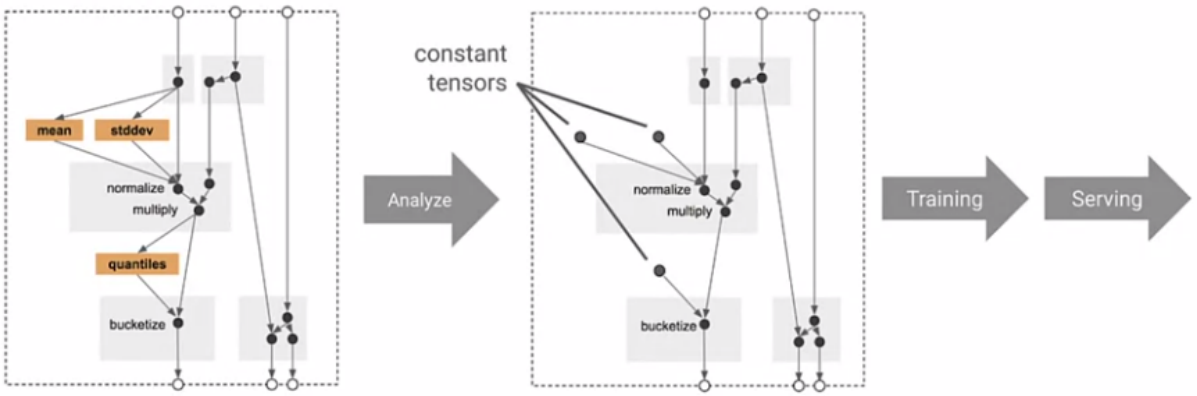
Analyzers: Full path over our dataset in order to collect content for feature engineering



They behave like TensorFlow Ops, but run only once during training

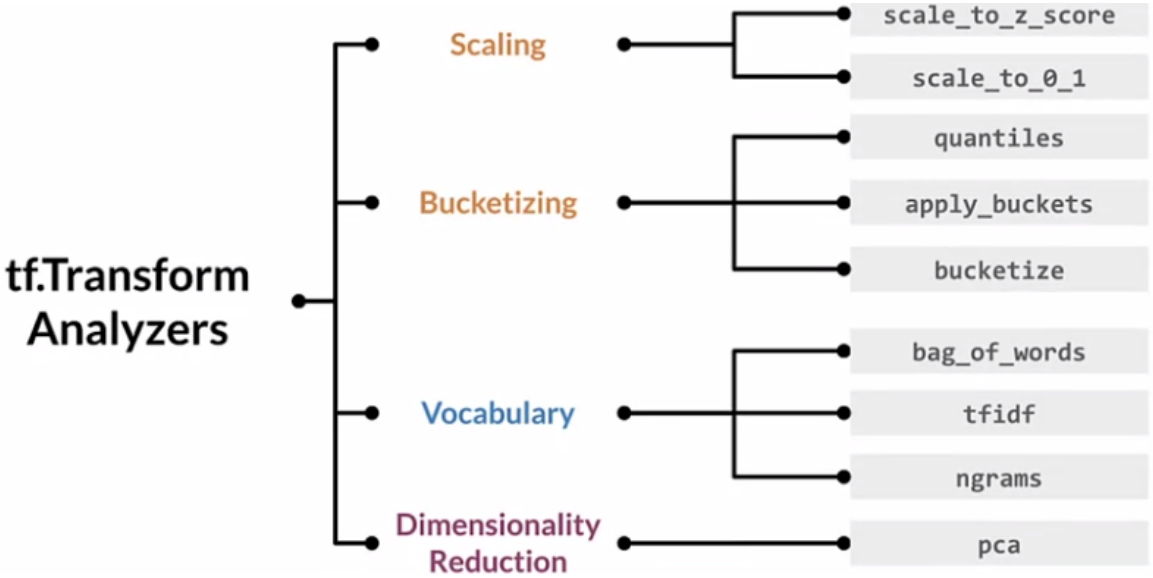
For example:
tft.min *computes the minimum of a tensor over the training dataset*

This generate a graph (same code) for training and serving without passing through all the dataset:



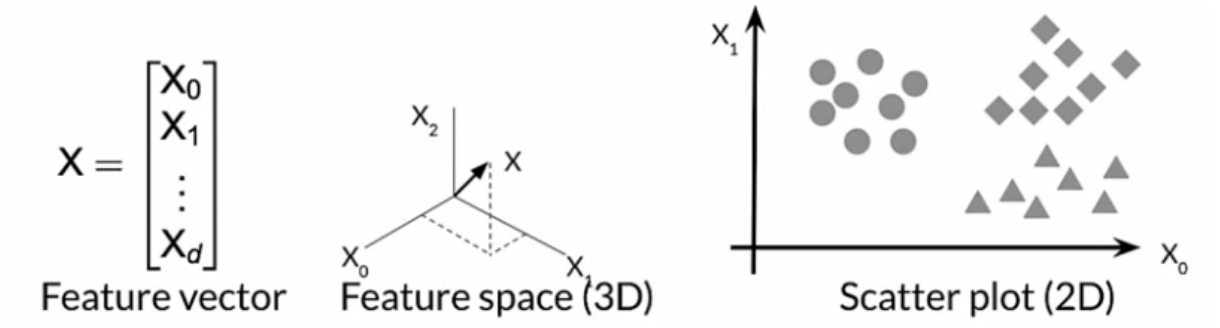
Benefits:

- Emitted Graph holds all necessary constants and transformations
- Focus on data preprocessing only at training time
- Works in-line during both training and serving
- No need for preprocessing code at serving time
- Consistently applied transformations irrespective of deployment platform



Feature Space:

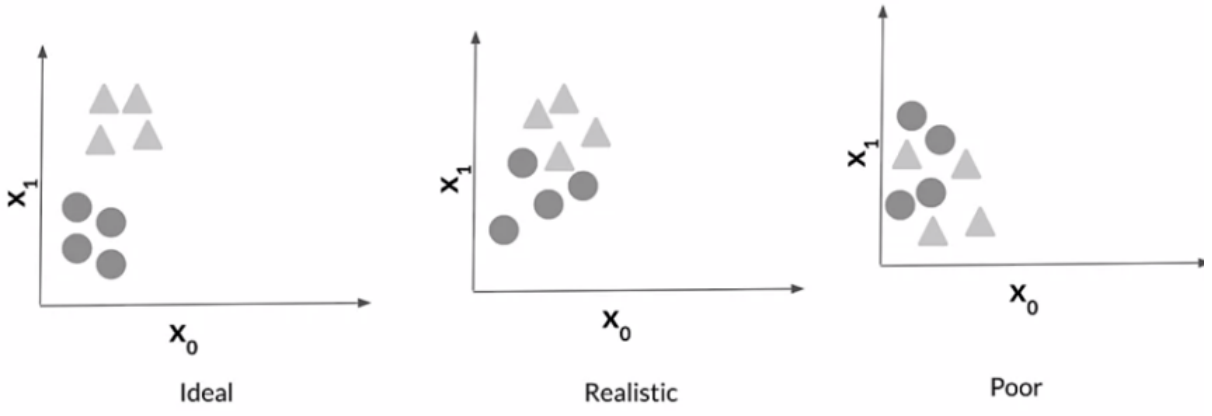
- N dimensional space defined by your N features
- Not including the target label



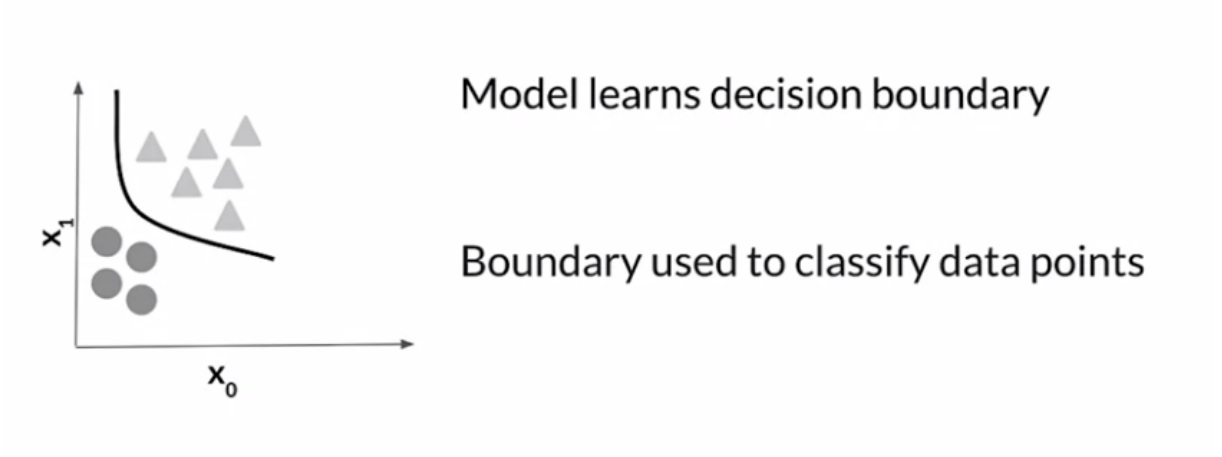
3D Feature Space

No. of Rooms X_0	Area X_1	Locality X_2	Price Y
5	1200 sq. ft	New York	$40,000
6	1800 sq. ft	Texas	$30,000

Classification feature space:



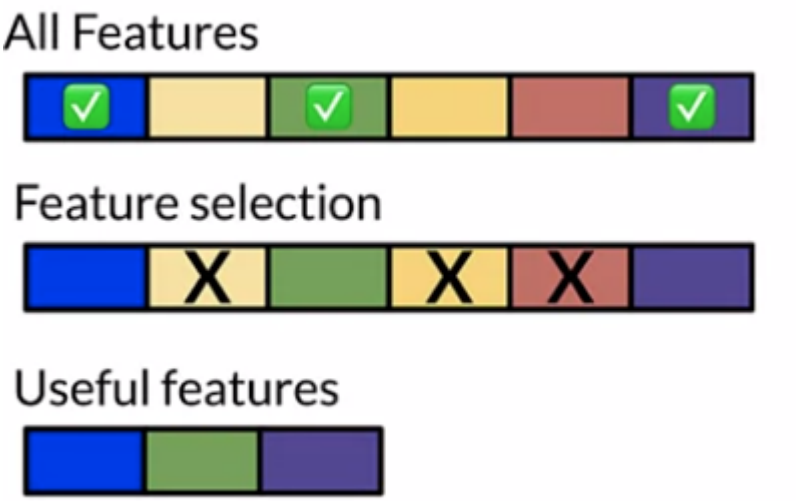
Decision Boundary:



Feature space coverage

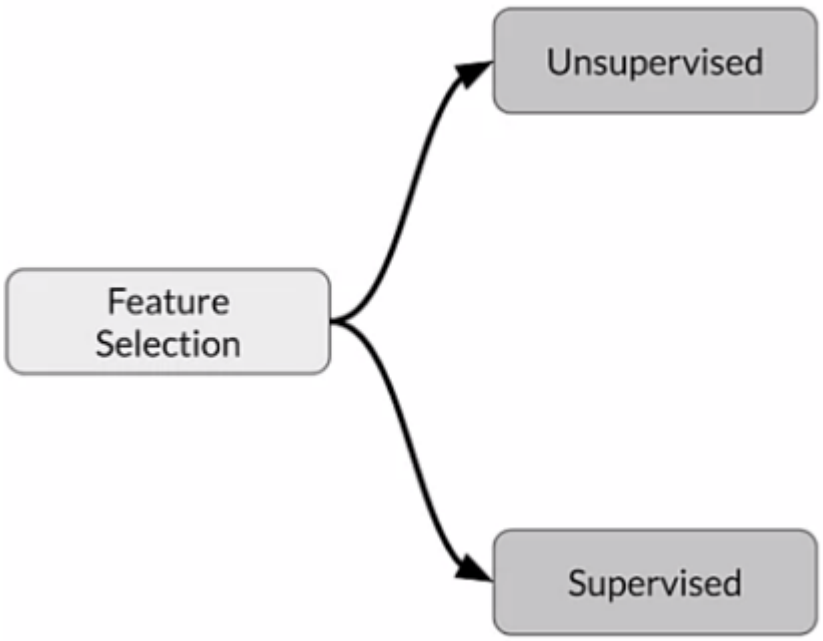
- Train/Eval datasets representative of the serving dataset
    - Same numerical ranges
    - Same classes
    - Similar characteristics for image data
    - Similar vocabulary, synstas, and semantics for NLP problems.
- Data affected by: seasonality, trend, drift.
- Serving data: new values in features and labels.
- Continuous monitoring, key for success!

Feature selection:

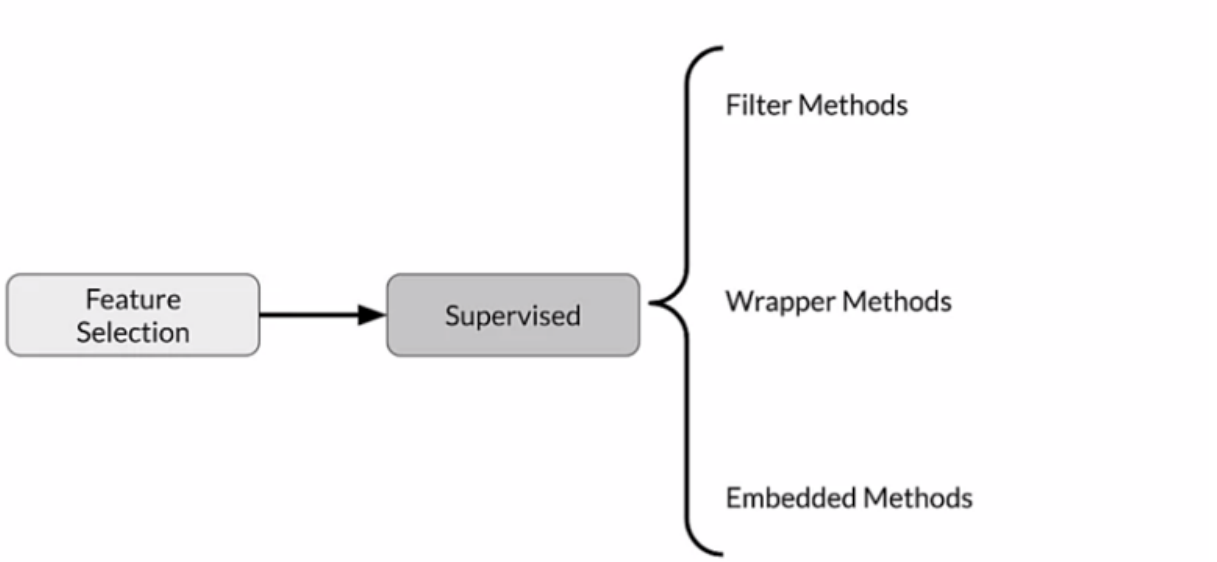


- Identify features that best represent the relationship.
- Remove features that don’t influence the outcome
- Reduce the size of the feature space
- Reduce the resource requirements and model complexity

Methods:



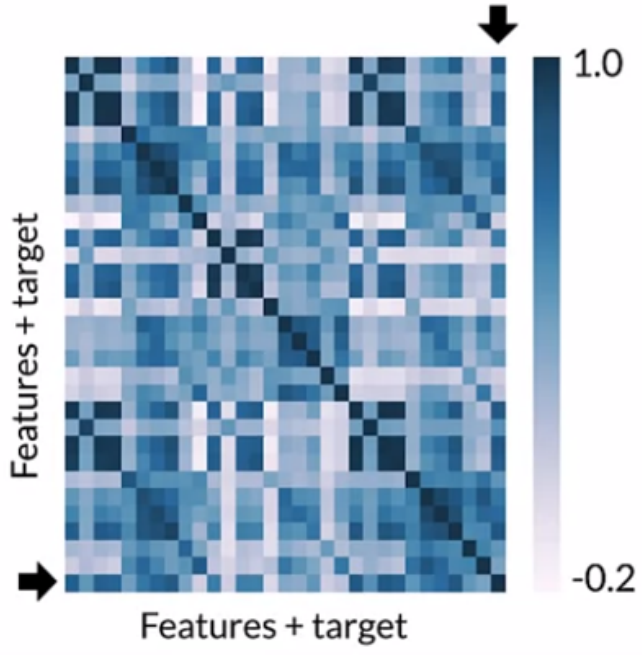
- Unsupervised:
  - Features-Target variables relationships not considered
  - Removes redundant features (correlation)
- Supervised:
  - Uses features-target variable relationships
  - Select those contributing the most



Filter Methods: A supervised method for doing feature selection, primarily we use correlation to look for the features that have correlation with our target.

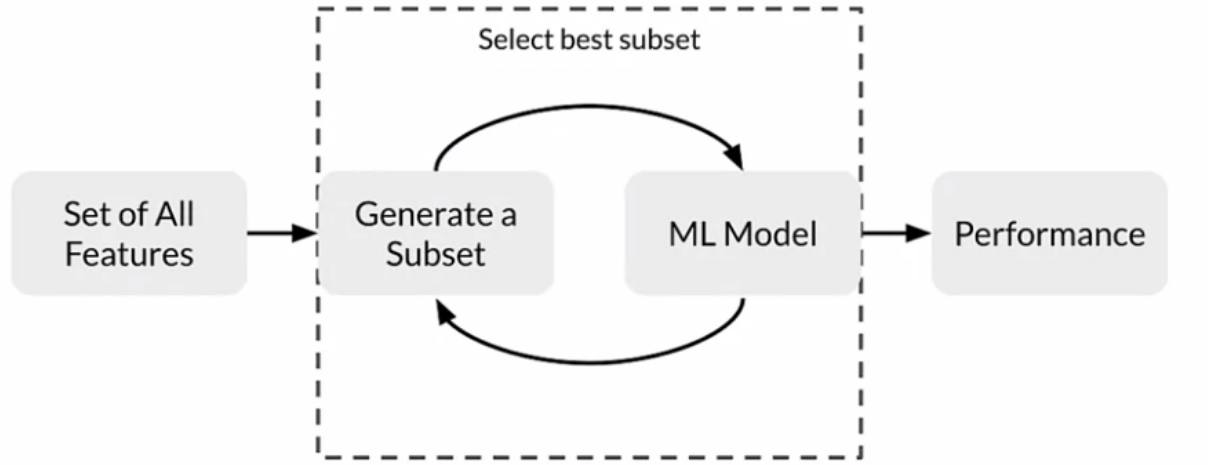
- Correlated features are usually redundant
    - Remove then
- Popular filter:
    - Pearson correlation:
        - Between features, and between the features and the label
    - Univariate Feature Selection

One way to visualize is using correlation matrix plot:



- Shows how features are related:
    - To each other (BAD)
    - And with target variable (GOOD)
- Falls in the range [-1, 1]
    - 1 High positive correlation
    - -1 High negative correlation
- Pearson's correlation: Linear relationship
- Kendall Tau Rank Correlation coefficient: Monotonic relationship & small sample size
- Spearman's Rank Correlation Coefficient: Monotonic relationship.
- Mutual information
- F-Test
- Chi-Squared test

Wrapper Methods: Supervised method, used with the model. Kind of a search method against the features: Forward, Backward or recursive elimination (Feature) For this we use random forest sklearn.



- Forward Selection: Iterative greedy method
    - Starts with 1 feature
    - Evaluate model performance when adding each of the additional features, one at a time
    - Add next feature that gives the best performance
    - Repeat until there is no improvement
- Backward Selection:
    - Start with all features
    - Evaluate model performance when removing each of the included features, one at a time
    - Repeat until there is no improvement
- Recursive Selection
    - Select a model to use for evaluating feature importance
    - Select the desired number of features
    - Fit the model
    - Rank features by importance
    - Discard least important features
    - Repeat until the desired number of features remains

Embed Methods: Supervised method, L1 regularization to get features, Feature importance, are intrinsic with the model. Used random forest sklearn (models type that have feature importance).

- Assigns scores for each feature in data
- Discard features scored lower by feature importance

Week 2: Feature Engineering, Transformation and Selection

If you wish to dive more deeply into the topics covered this week, feel free to check out these optional references. You won't have to read these to complete this week's practice quizzes.

Mapping raw data into feature

Feature engineering techniques

Scaling

Facets

Embedding projector

Encoding features

TFX:

1. https://www.tensorflow.org/tfx/guide#tfx_pipelines
2. https://ai.googleblog.com/2017/02/preprocessing-for-machine-learning-with.html

Breast Cancer Dataset