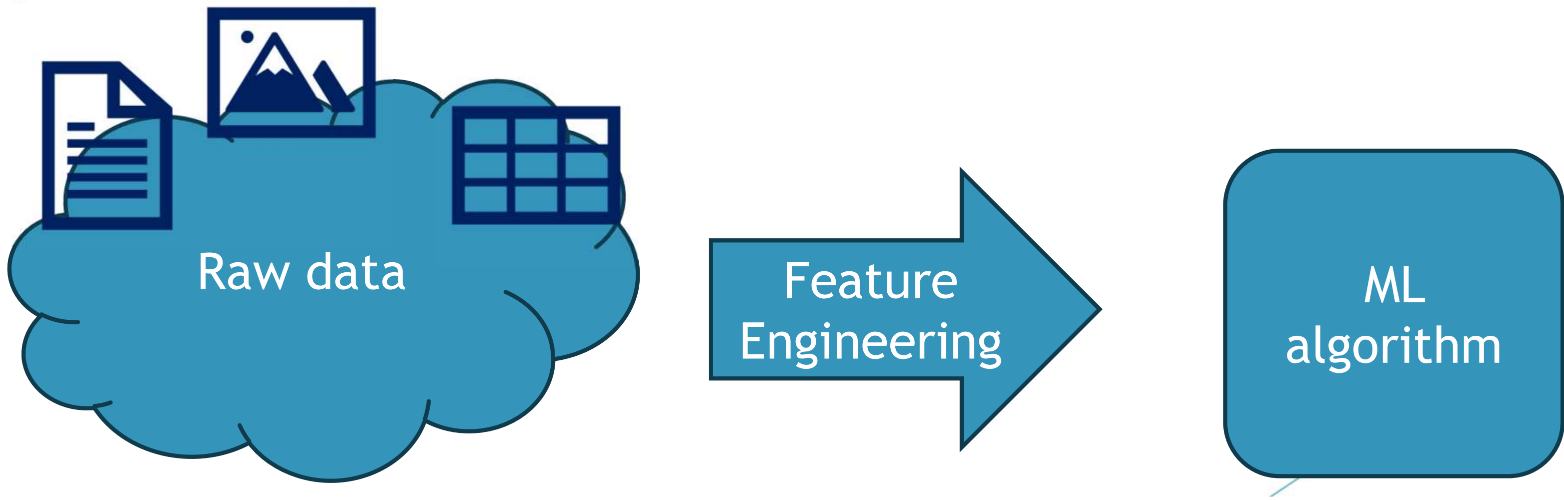
Feature Engineering

Maryam Abdolali

KNTU, Fall 2024

Introduction

- Feature Engineering: plays a crucial role in training the machine learning models
- *Feature engineering:*
  - process of *selecting*, *transforming* and *creating* relevant input variables (features) from raw data
- *Why:*
  - **Improve model performance**
  - **Lessen computational costs**
  - **Improve model interpretability**



Component Processes of Feature Engineering

Feature Engineering

Feature Creation

combining or deriving information from existing features

Feature Transformation

Modifying existing features to make them more useful

Feature Extraction

Creating new features by extracting meaningful patterns or reducing dimensionality

Feature Selection

Choosing the most relevant features to reduce overfitting and improve model performance

Component Processes of Feature Engineering

- **Feature Creation**
- *combining* or *deriving* information from existing features
  - Use domain knowledge, Multiply or combine features
  - Example: RFM (Recency, Frequency, Monetary)
    - Recency: Time since the last purchase
    - Frequency: Number of purchases in a given time frame
    - Monetary: Total amount spent by the customer

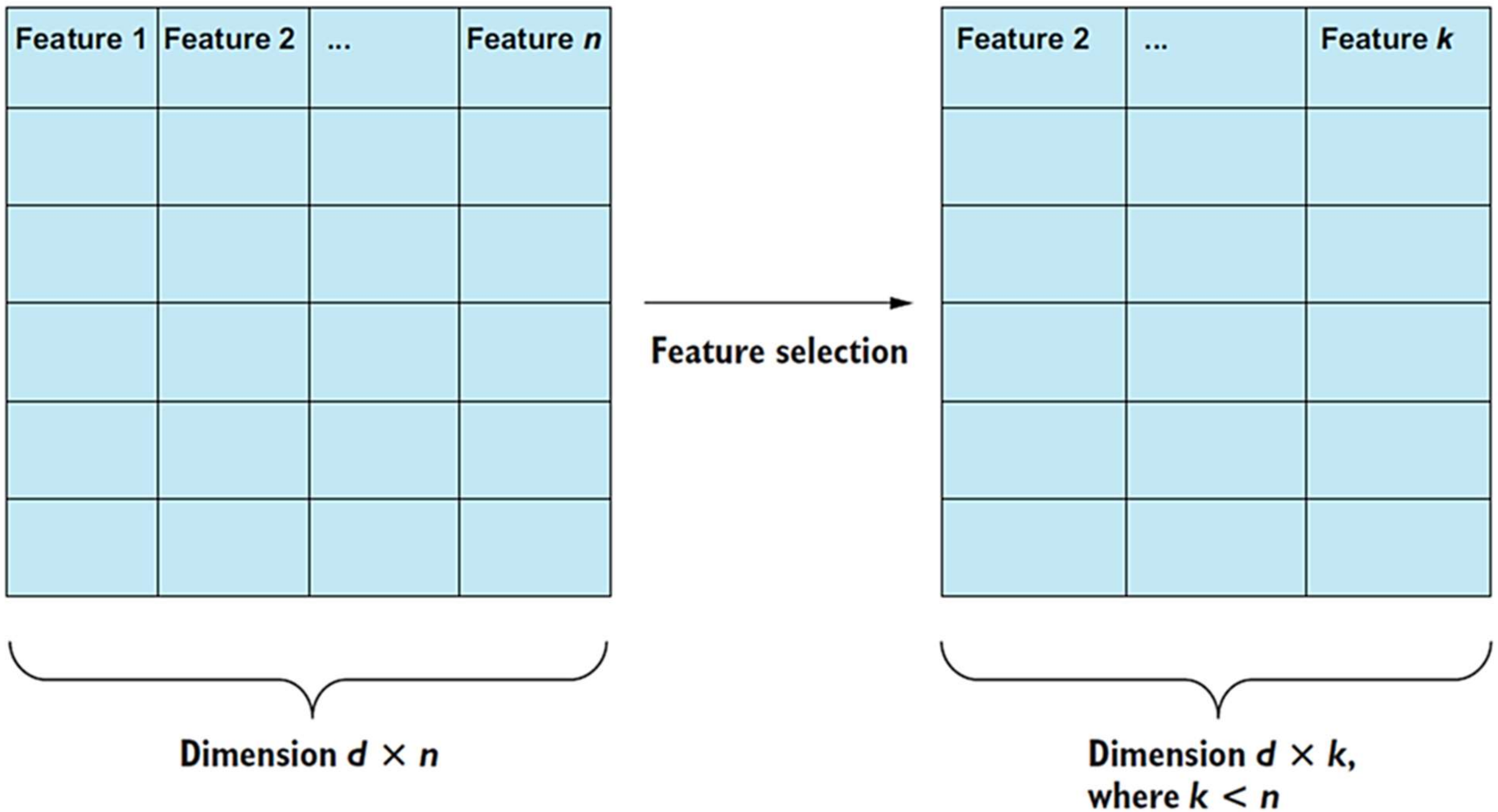
- **Feature Transformation**
- *Modifying existing features* to make them more useful
- Techniques, includes:
  - *Scaling and Normalization*
  - *Log and Power Transformations*
  - *Encoding Categorical Features*
  - *Binning and Discretization*

Component Processes of Feature Engineering

- **Feature Extraction**
- Creating new features by extracting meaningful patterns or *reducing dimensionality*
  - Techniques:
    - Principal Component Analysis (PCA): Dimensionality reduction through linear projections.
    - ~~t-SNE and UMAP: Visualization and structure discovery~~

- **Feature Selection**
- **Choosing** the *most relevant features* to reduce overfitting and improve model performance.
  - Techniques:
    - Filter Methods: Variance Threshold, SelectKBest (ANOVA, Chi-Square).
    - Wrapper Methods: Recursive Feature Elimination (RFE), Sequential Feature Selector.
    - Embedded Methods: Lasso regression (L1 regularization), tree-based models (Random Forest).

Feature Selection



selecting the best subset of existing features

Filter Methods

- **Independent** of the machine learning model
- Simple, fast, and useful as a preprocessing step

In Scikit-Learn: **SelectKBest**

selects the top k features based on a scoring function that evaluates the statistical relationship between each feature and the target variable. It does not consider interactions between features.

1. Variance Threshold
   - Removes features with low variance, assuming that features with low variance are less likely to be useful for distinguishing classes.
2. Statistical tests
   - Pearson correlation
     - removing features that are less relevant to the target variable or are highly correlated with other features
   - ~~ANOVA (Analysis of Variance)~~
   - ~~Chi-square test for categorical variables~~

Example for filter method

- **Data:**

X1	X2	X3	Target
1	2	5	10
2	2	4	20
3	5	4	30
4	2	2	40
5	3	-5	50

- **Using correlation coefficients** to score the relevance of each feature with the target variable

$$r_{xy} = \frac{Cov(X,y)}{\sigma_X \sigma_y}$$

$$Cov(X,y) = \frac{\sum (X_i - \bar{X})(y_i - \bar{y})}{n}$$

Solution

- For $X1$:
  - $Mean(X_1) = \frac{1+2+3+4+5}{5} = 3,\ Mean(y) = 30$
  - $Cov(X, y) = \frac{\sum(X_i - \bar{X})(y_i - \bar{y})}{n}$
  - $Cov(X1, y) = \frac{(1-3)(10-30)+(2-3)(20-30)\ \ (3-3)(30-30)+(4-3)(40-30)+(5-3)(50-30)}{5} = 20$
  - $\sigma_X = \sqrt{\frac{\sum(X_i - \bar{X})^2}{n}},\ \sigma_y = \sqrt{\frac{\sum(y_i - \bar{y})^2}{n}}$
  - $\sigma_{X1} = \sqrt{\frac{(-2)^2+(-1)^2+0^2+1^2+2^2}{5}} = 1.41,\ \sigma_y = \sqrt{\frac{(-20)^2+(-10)^2+0^2+10^2+20^2}{5}} = 14.14$
  - $corr(X1, y) = \frac{20}{1.41 * 14.14} = 1.0\ \rightarrow score\ (X1) = 1$
- Similarly for X2: $corr(X2, y) = 0.24 \rightarrow score\ (X2) = 0.24$
- For X3: $corr(X3, y) = -0.86 \rightarrow score\ (X3) = 0.86$

❑ E.g., Using Filter methods we select $X1$, $X3$ as “top two features” among the three features.

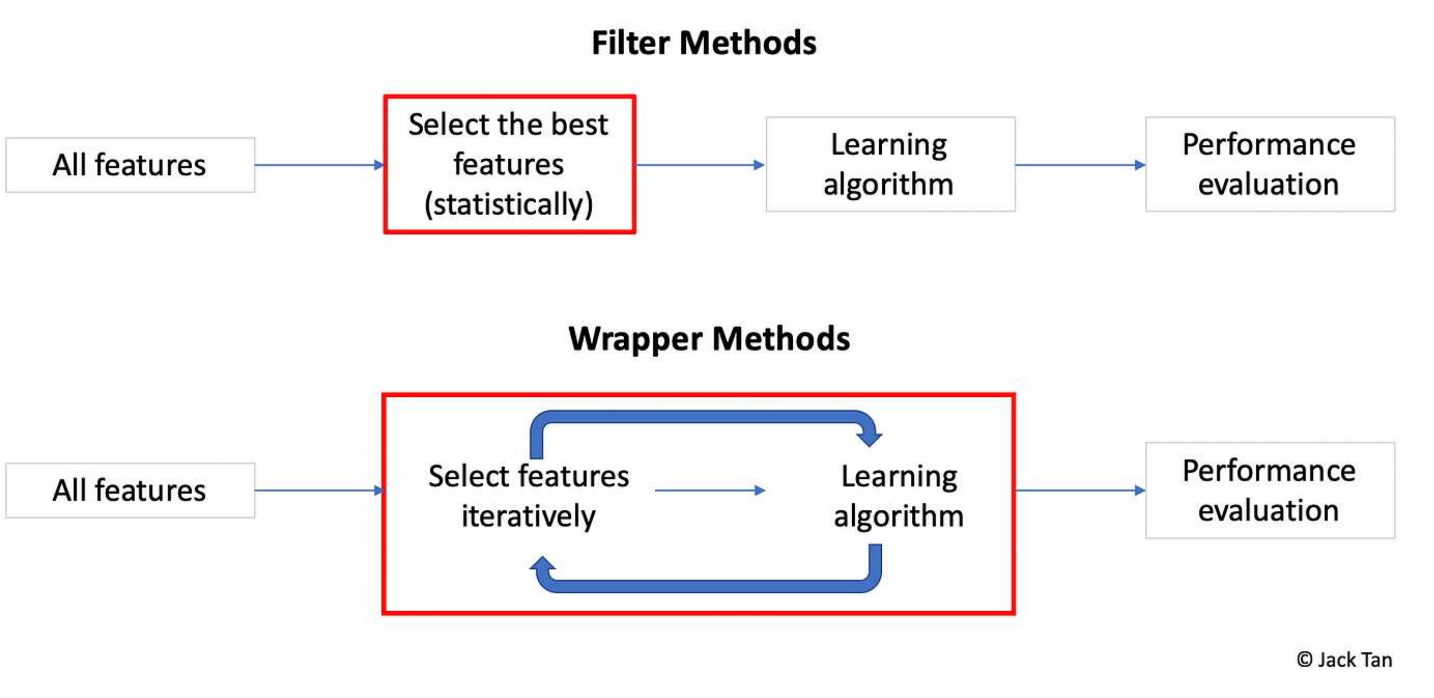
```python
import numpy as np
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
X = np.array([[1, 2, 3, 4],
              [2, 4, 6, 8],
              [3, 6, 9, 12],
              [4, 8, 12, 16],
              [5, 10, 15, 20]])
y = np.array([10, 20, 30, 40, 50])
X = X + np.random.normal(0, 0.5, X.shape)

k_best = SelectKBest(score_func=f_regression, k=2)
X_selected = k_best.fit_transform(X, y)

model = LinearRegression()
model.fit(X_selected, y)
y_pred = model.predict(X_selected)
mse = mean_squared_error(y, y_pred)
print("Mean Squared Error:", mse)
```

Select k best features independently from the model

Fit the model on selected features

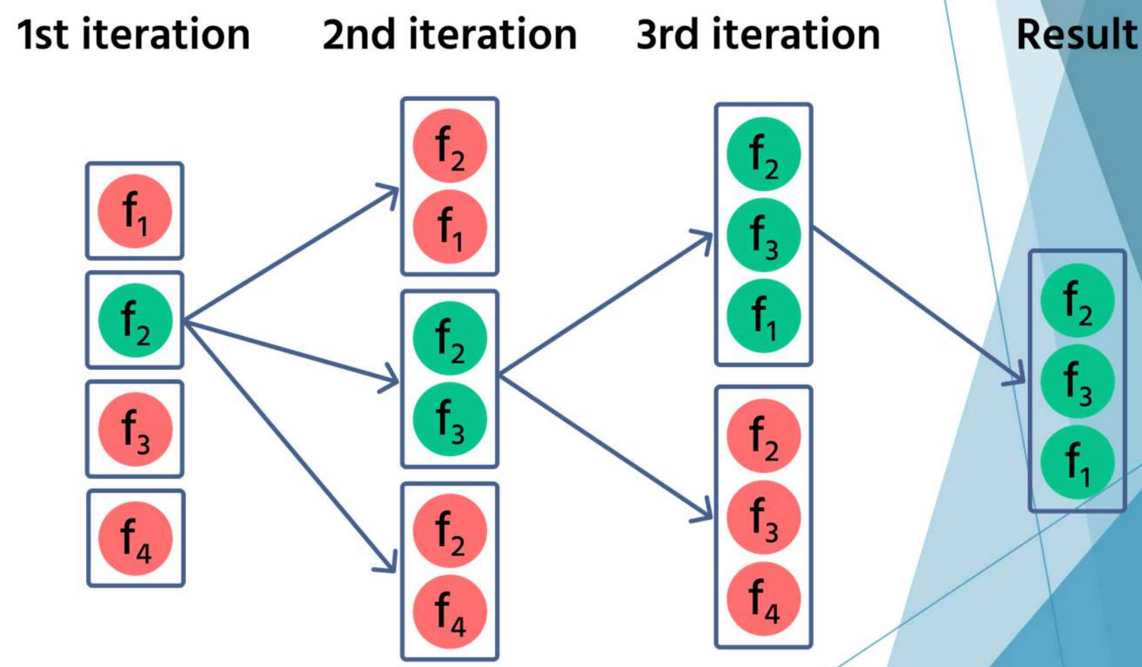
Wrapper Methods

- **Forward Selection:**
  - Starts with no features and iteratively adds the most significant feature based on model performance. Stops when adding features no longer improves performance.
- **Backward Elimination:**
  - Starts with all features and iteratively removes the least significant feature. Stops when removing features reduces model performance.



Forward Selection

- **Initialize:** Start with an *empty feature set* and prepare to *add features one by one*.
- **Model Training and Evaluation:** For *each feature*, train the model with the current set of selected features and evaluate its performance.
- **Feature Selection:** Add the feature that most improves model performance.
- **Stopping Condition:** Repeat until the stopping criterion is reached, such as a maximum feature count or minimal improvement in performance.

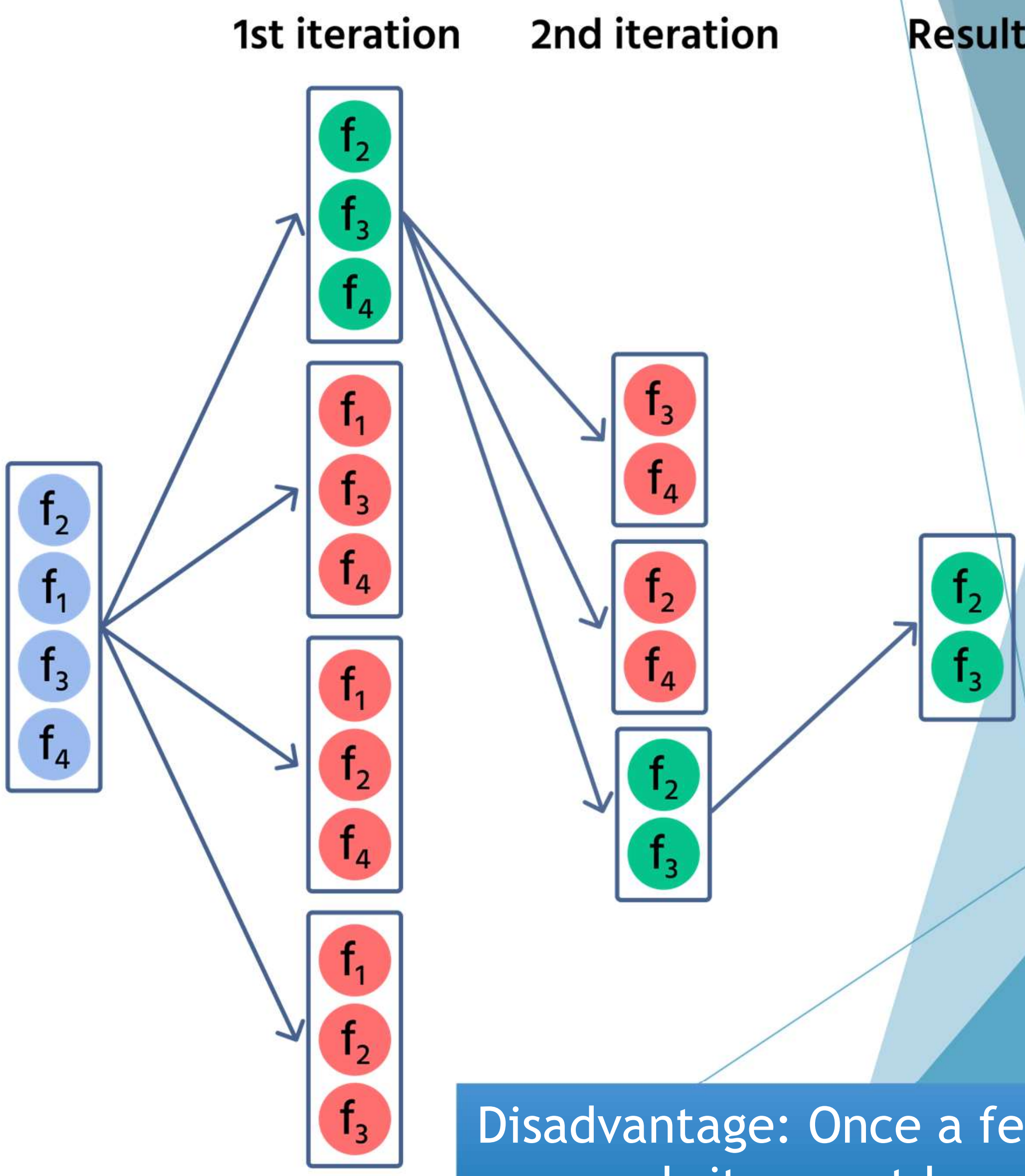


Disadvantage: Once a feature is added, it cannot be discarded.

Image from https://codefinity.com/blog/Sequential-Backward-and-Forward-Selection

Backward Selection

- **Initialize:** Start with *all features* and prepare to remove them one by one.
- **Model Training and Evaluation:** For each feature, *train* the model with the current set of selected features and *evaluate* its performance.
- **Feature Removal:** Remove the feature whose absence causes the *least performance drop*.
- **Stopping Condition:** Repeat until the stopping criterion is reached, such as a *minimum number of features* or *significant performance degradation*.



Disadvantage: Once a feature is removed, it cannot be added.

https://codefinity.com/blog/Sequential-Backward-and-Forward-Selection

Example

- You want to predict house prices (y) based on several features:
  - Size (square feet)
  - Number of Bedrooms
  - Age of the House
- Split into:
- **Training set**: Rows 1-3.
- **Validation set**: Rows 4-5.

	Size	Bedrooms	Age	price
Train	2000	3	10	400
Train	1500	2	20	250
Train	1800	4	15	350
Test	1200	2	30	200
Test	2500	4	5	500

- select the most informative features to predict house prices using **SFS** with a simple linear regression model

Feature 1: Size

- Training data:

$$X_{train} = \begin{bmatrix} 2000 \\ 1500 \\ 1800 \end{bmatrix}, \qquad y_{train} = \begin{bmatrix} 400 \\ 250 \\ 350 \end{bmatrix}$$

- Validation data:

$$X_{Val} = \begin{bmatrix} 1200 \\ 2500 \end{bmatrix}, \qquad y_{train} = \begin{bmatrix} 200 \\ 500 \end{bmatrix}$$

- Fit Linear Regression:

$$y = \beta_0 + \beta_1 . Size$$

- Coefficient: 0.1.Intercept: 50.
- Validation Predictions: [170, 300].Validation MSE: 2900

Feature 2: Bedrooms

- Training data:

$$X_{train} = \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix}, \qquad y_{train} = \begin{bmatrix} 400 \\ 250 \\ 350 \end{bmatrix}$$

- Validation data:

$$X_{Val} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \qquad y_{train} = \begin{bmatrix} 200 \\ 500 \end{bmatrix}$$

- Fit Linear Regression:

$$y = \beta_0 + \beta_1 . Bedrooms$$

- Coefficient: 90.Intercept: 50.
- Validation Predictions: [230, 410].Validation MSE: 3100.

Feature 3: Age

- Training data:

$$X_{train} = \begin{bmatrix} 10 \\ 20 \\ 15 \end{bmatrix}, \qquad y_{train} = \begin{bmatrix} 400 \\ 250 \\ 350 \end{bmatrix}$$

- Validation data:

$$X_{Val} = \begin{bmatrix} 30 \\ 5 \end{bmatrix}, \qquad y_{train} = \begin{bmatrix} 200 \\ 500 \end{bmatrix}$$

- Fit Linear Regression:

$$y = \beta_0 + \beta_1 . Age$$

- Coefficient: -5.Intercept: 400.
- Validation Predictions: [250, 375].Validation MSE: 3250.

- **Select feature Size**

Features Size and Bedrooms

$$X_{train} = \begin{bmatrix} 2000 & 3 \\ 1500 & 2 \\ 1800 & 4 \end{bmatrix}, \quad y_{train} = \begin{bmatrix} 400 \\ 250 \\ 350 \end{bmatrix}$$

$$y = \beta_0 + \beta_1 . Size + \beta_2 . Bedrooms$$

y = 50 + 0.15 * Size + 10 * Bedrooms

MSE = 1862.5

Features Size and Age

$$X_{train} = \begin{bmatrix} 2000 & 10 \\ 1500 & 20 \\ 1800 & 15 \end{bmatrix}, \quad y_{train} = \begin{bmatrix} 400 \\ 250 \\ 350 \end{bmatrix}$$

$$y = \beta_0 + \beta_1 . Size + \beta_2 . Age$$

y = 78 + 0.16 * Size − 2 * Age

MSE = 562

Select features Size & Age

Feature Selection using SFS (SBS) in Scikit-Learn

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X, y = make_classification(n_samples=100, n_features=10, n_informative=5, n_redundant=2,
random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()

sfs_forward = SequentialFeatureSelector(
    model,
    n_features_to_select=5,
    direction='forward',
    scoring='accuracy',
    cv=5
)

X_train_forward=sfs_forward.fit_transform(X_train, y_train)
X_test_forward = sfs_forward.transform(X_test)

model_forward = LogisticRegression()
model_forward.fit(X_train_forward, y_train)
y_pred_forward = model_forward.predict(X_test_forward)
accuracy_forward = accuracy_score(y_test, y_pred_forward)
print("Test accuracy (Forward Selection):", accuracy_forward)
```

Select features for both train & test using SFS

Fit the model on the selected features for train data

Recursive Feature Elimination (RFE)

- **Initialize:** Start with all features and define a model.
- **Model Training and Evaluation:** Train the model with the current set of features and evaluate performance.
- **Feature Ranking and Removal:** Rank features by importance (e.g., coefficients or feature importance) and remove the least important feature.
- **Repeat:** Refit the model with the remaining features and repeat until the desired number of features is reached or performance degrades significantly.

The difference between RFE & SBS:

- RFE: Train the model → Rank features by importance → Remove **the least important feature** → Repeat.
- SBS: Train the model → Evaluate performance with each feature removed → Remove **the feature that causes the least performance drop** → Repeat.

Note:

- ✓ Models such as Linear Regression, Logistic Regression return "coefficients"
- ✓ Models such as Decision Trees / Random Forest return "feature importance"

can be used with any estimator that exposes the "coef_" or "feature_importances_" attributes, such as logistic regression.

Example for RFE

- **Fit a Linear Regression Model:** We first fit a linear regression model using all three features.

$$Price\ (in\ thousands) = \beta_0 + \beta_1 X1 + \beta_2 X2 + \beta_3 X3$$

$$price = 326 + 94.81\ (size) - 22.51\ (bedrooms) + 7.70\ (bathrooms)$$

- Based on the magnitude of the coefficients, we rank the features in order of importance
  - Size -> bedrooms -> bathrooms
- **Eliminate the Least Important Feature:** The least important feature, **X3 (Number of Bathrooms)**, is removed from the model.

Size	# bedrooms	# bathrooms	price
1500	3	2	300
1800	4	2.5	350
1200	2	1	250
2200	4	3	450
1400	3	2	280

Size	# bedrooms	# bathrooms	price
-0.3	-0.25	-0.125	300
0.45	1.0	0.5	350
-1.05	-1.5	-1.375	250
1.45	1.0	1.125	450
-0.55	-0.25	-0.125	280

-cont-

- Fit the Model Again with Remaining Features (X1 and X2)

$$price\ (in\ thousands) = \beta_0 + \beta_1 X1 + \beta_2 X2$$
$$price = 326 + 97.54\ (size) - 18.21\ (bedrooms)$$

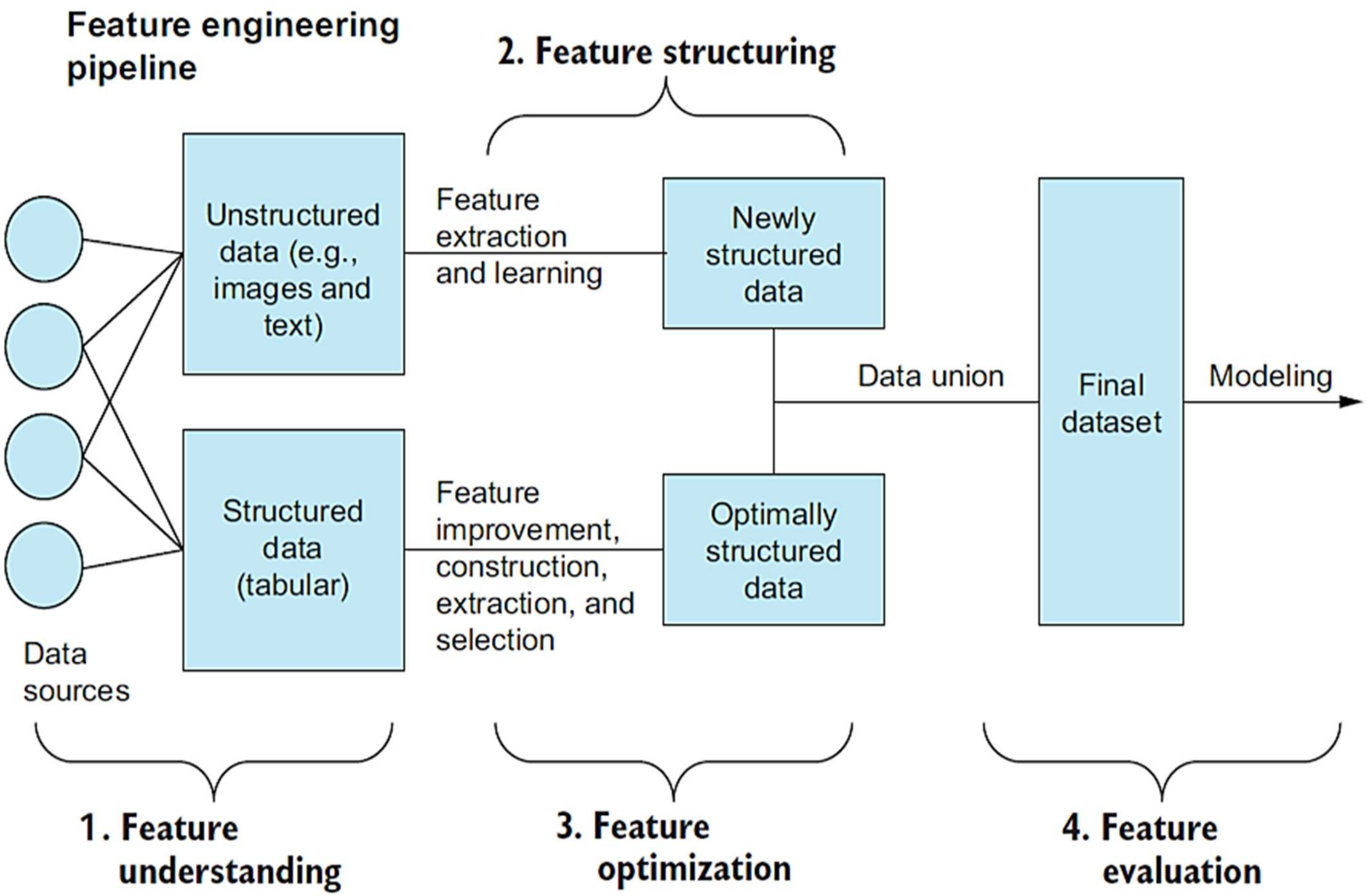
- Based on the magnitude of the coefficients, we rank the features in order of importance
  - Size -> bedrooms
- **Eliminate the Least Important Feature:** The least important feature, **X2 (Number of Bedrooms)**, is removed from the model.

Feature Selection using RFE in Scikit-Learn

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X, y = make_classification(n_samples=100,n_features=10, n_informative=5, n_redundant=2,
random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()

rfe = RFE(estimator=model, n_features_to_select=5)

X_train_rfe = rfe.fit_transform(X_train, y_train)
X_test_rfe = rfe.transform(X_test)

model.fit(X_train_rfe, y_train)
y_pred = model.predict(X_test_rfe)
accuracy = accuracy_score(y_test, y_pred)

print("Selected features:", rfe.support_)
print("Ranking of features:", rfe.ranking_)
print("Test accuracy with selected features:", accuracy)
```

Select features for both train & test using RFE

Fit the model on the selected features for train data

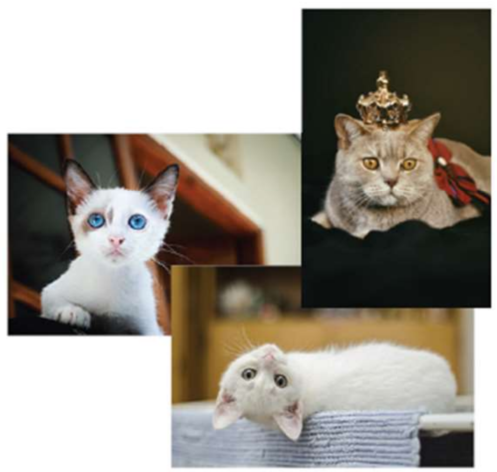

Feature engineering pipeline
2. Feature structuring
Unstructured data (e.g., images and text)
Feature extraction and learning
Newly structured data
Data union
Final dataset
Modeling
Structured data (tabular)
Feature improvement, construction, extraction, and selection
Optimally structured data
Data sources
1. Feature understanding
3. Feature optimization
4. Feature evaluation

Feature Engineering for Unstructured data

- Raw data, such as *text, audio, images, and videos*, must be transformed into **numerical** vector representations to be processed by any ML algorithm. This process, which we will refer to as ***feature structuring***, can be done through extraction techniques

I can't login to my account
I need to speak to a human
Help!!!!111!!!
Nothing works...

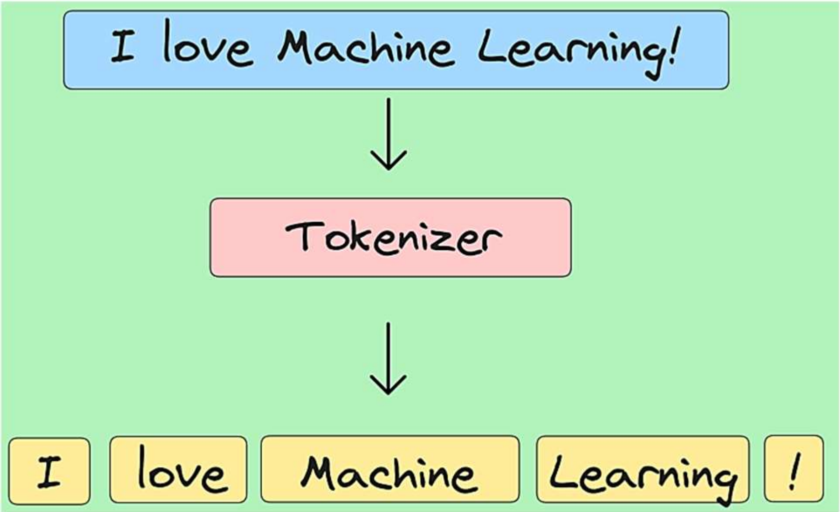
I	Can't	...	Help
0	1	...	0
0	0	...	9
2	0	...	1
5	0	...	1
0	1	...	1



Feature 1 (e.g., has crown)	Feature 2 (e.g., is laying down)	...	Feature *n*
0.324	1	...	0
0	0	...	9.234
2	0	...	.421
8.4	0	...	.961
0	1	...	1

Feature Engineering for Text

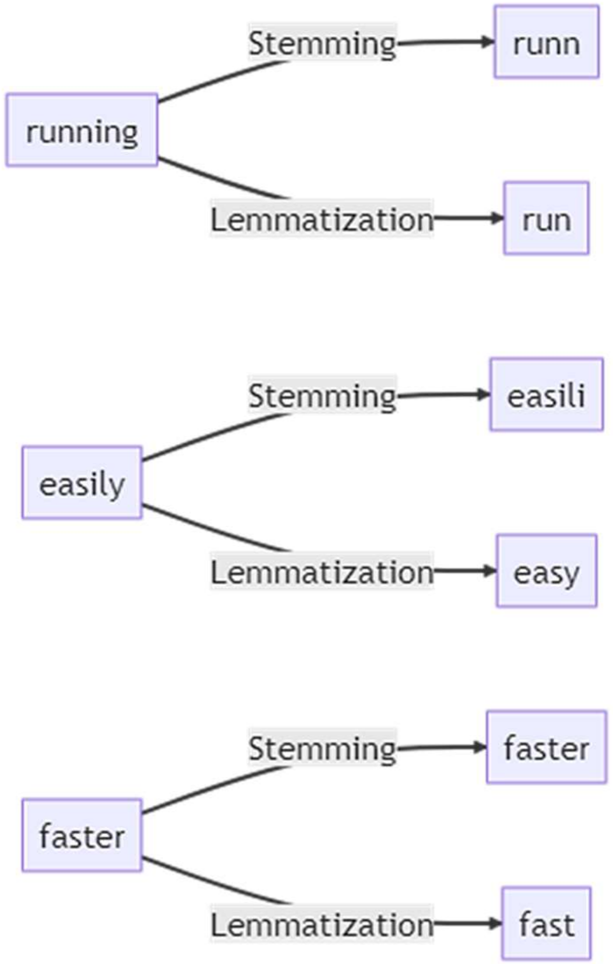
- **Tokenization**
  - splitting the text into smaller units (tokens), such as words or subwords



- **Stopword Removal**
  - Stopwords are common words like "the", "is", "in", "on", "a", etc., that don't carry much meaning and can be safely ignored

-cont-

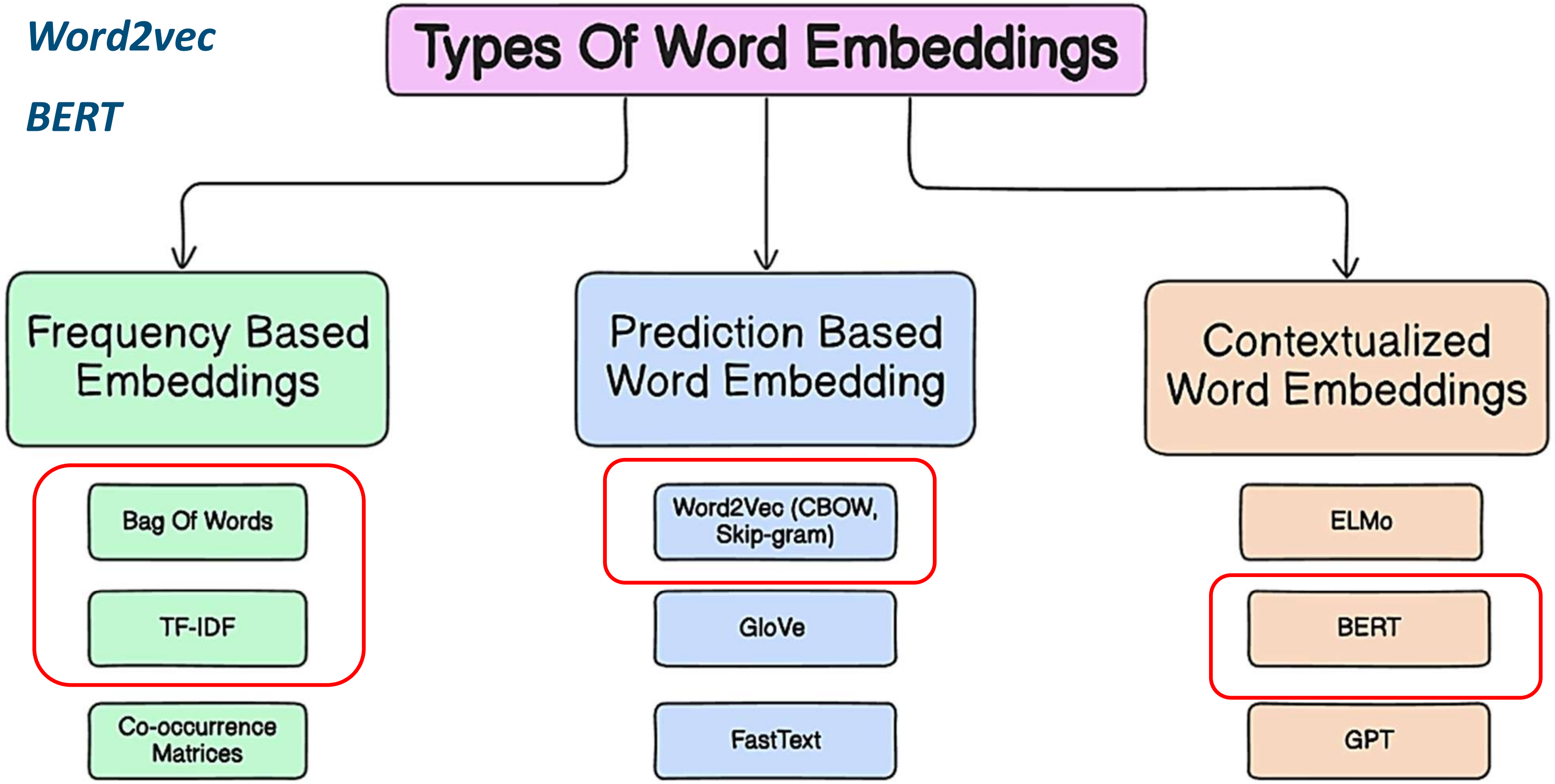
- **Stemming**
  - reduces words to their root form by chopping off prefixes and suffixes
- **Lemmatization**
  - reduces words to their base or dictionary form



Vectorizer for Tokens

Goal: Convert processed tokens to vectors of numbers

- ***Bag of words***
- ***TF-IDF vectorization***
- ***Word2vec***
- ***BERT***



Bag of Words

m1: A word of text.
m2: A word is a token.
m3: Tokens and features.
m4: Few features of text.

Training data

Bag of words

word of text
a
word a is a
token
tokens features
and
few features
of text

Tokens

One feature per unique token

x_1	a
x_2	word
x_3	of
x_4	text
x_5	is
x_6	token
x_7	tokens
x_8	and
x_9	features
x_{10}	few

Features

Bag of Words: Example

test1: Some features for a text example.

m1: A word of text.
m2: A word is a token.
m3: Tokens and features.
m4: Few features of text.

x_1	a
x_2	word
x_3	of
x_4	text
x_5	is
x_6	token
x_7	tokens
x_8	and
x_9	features
x_{10}	few

Selected Features

	m1	m2	m3	m4
x_1	1	1	0	0
x_2	1	1	0	0
x_3	1	0	0	1
x_4	1	0	0	1
x_5	0	1	0	0
x_6	0	1	0	0
x_7	0	0	1	0
x_8	0	0	1	0
x_9	0	0	1	1
x_{10}	0	0	0	1

Training X

	test1
x_1	1
x_2	0
x_3	0
x_4	1
x_5	0
x_6	0
x_7	0
x_8	0
x_9	1
x_{10}	0

Test X

Use bag of words when you have a lot of data, can use many features

TF-IDF
Term Frequency – Inverse Document Frequency

- Instead of using binary: ContainsWord(<term>)
- Use numeric importance score TF-IDF:

Importance to Document

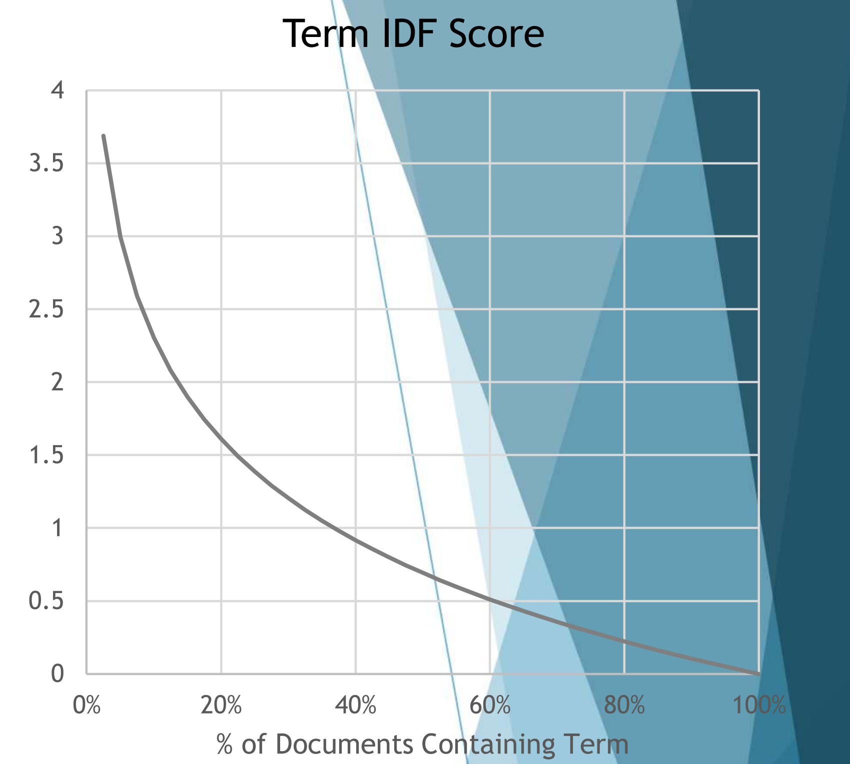
TermFrequency(<term>, <document>) =

% of the words in <document> that are <term>

Novelty across corpus

InverseDocumentFrequency(<term>, <documents>) =

log (# documents / # documents that contain <term>)



Words that occur in many documents have low score (x_i)

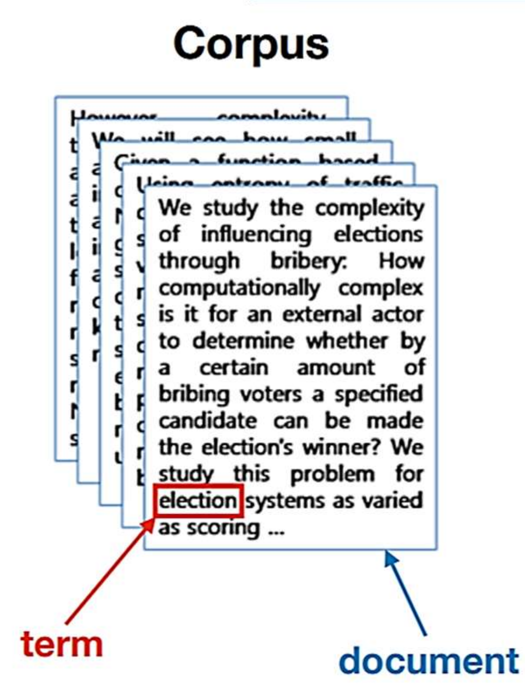
Message 1: “Nah I don't think he goes to usf”
Message 2: “Text FA to 87121 to receive entry”

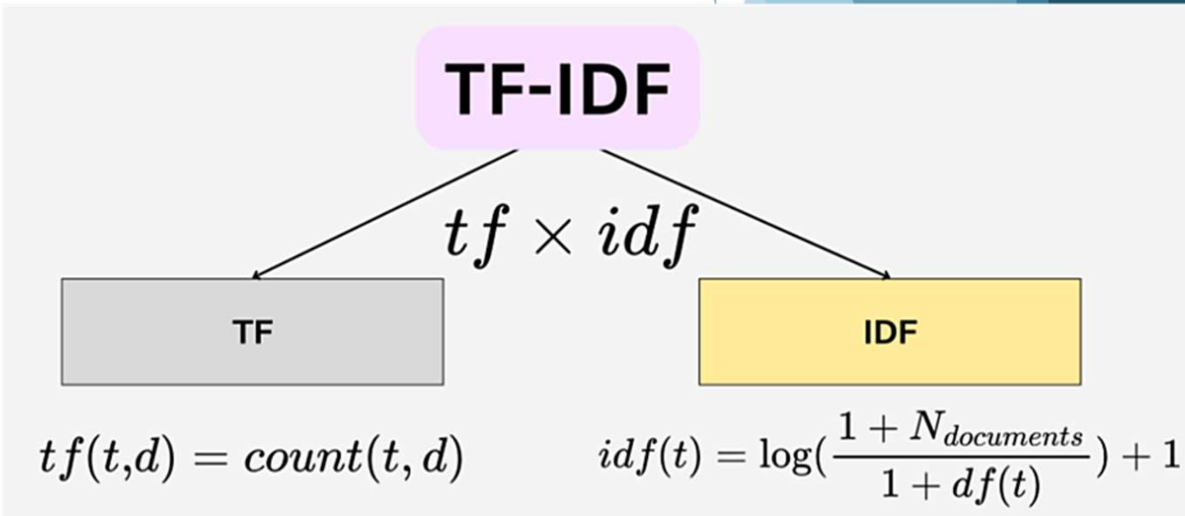
Message 2:

	Nah	I	don't	think	he	goes	to	usf	Text	FA	87121	receiv e	entry
BOW	0	0	0	0	0	0	1	0	1	1	1	1	1
TF-IDF	0	0	0	0	0	0	0	0	.099	.099	.099	.099	.099

Example Tf-Idf

- Dataset: Take the following four strings to be (very small) documents comprising a (very small) corpus:
  1. “The sky is blue.”
  2. “The sun is bright today.”
  3. “The sun in the sky is bright.”
  4. “We can see the shining sun, the bright sun.”
- Task: Filter out obvious stopwords, and determine the tf-idf scores of each term in each document.





Solution

- After stopword filtering:
  - (1) “sky blue”,
  - (2) “sun bright today”,
  - (3) “sun sky bright”,
  - (4) “can see shining sun bright sun”
- TF: Find doc-word matrix, then normalize rows to sum to 1

	blue	bright	can	see	shining	sky	sun	today
1	1	0	0	0	0	1	0	0
2	0	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1	0
4	0	1	1	1	1	0	2	0

→

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

Borrowed From https://www.cbrinton.net/ECE20875-2020-Spring/W10/ngrams.pdf

solution

- IDF: Find number of documents each word occurs in, then compute formula

	blue	bright	can	see	shining	sky	sun	today
1	1	0	0	0	0	1	0	0
2	0	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1	0
4	0	1	1	1	1	0	2	0
n_t	**1**	**3**	**1**	**1**	**1**	**2**	**3**	**1**

$N = 4$

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

$$\log_{10}\frac{4}{1} = 0.602 \qquad \log_{10}\frac{4}{3} = 0.125$$

Borrowed From https://www.cbrinton.net/ECE20875-2020-Spring/W10/ngrams.pdf

solution

tf

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

x

idf

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
  - Most important word for each document is highlighted

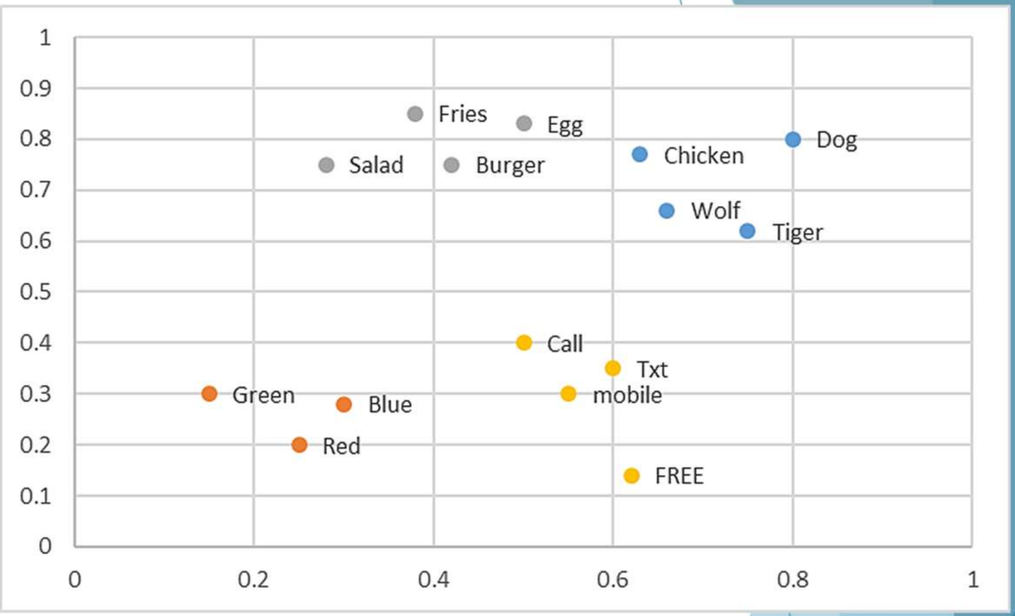
$\mathrm{tfidf} = \mathrm{tf} \cdot \mathrm{idf}$

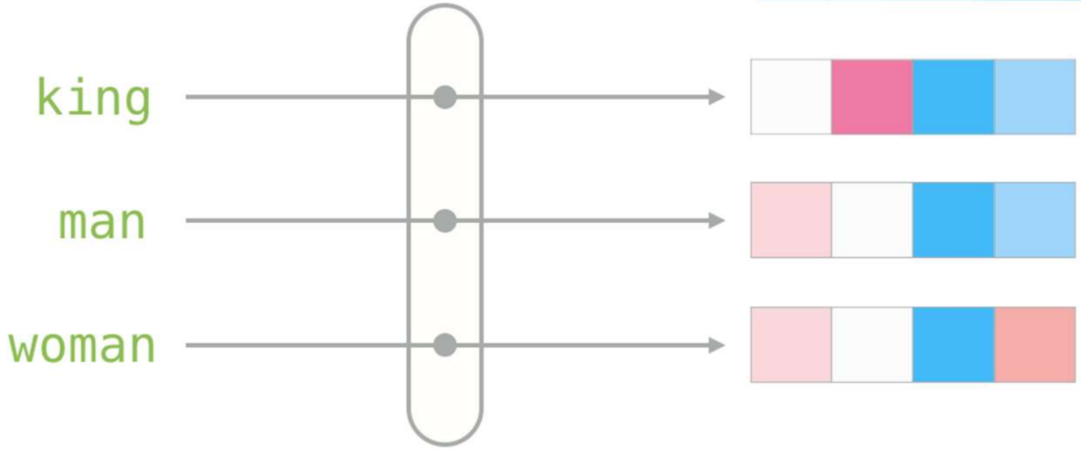
	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

Borrowed From https://www.cbrinton.net/ECE20875-2020-Spring/W10/ngrams.pdf

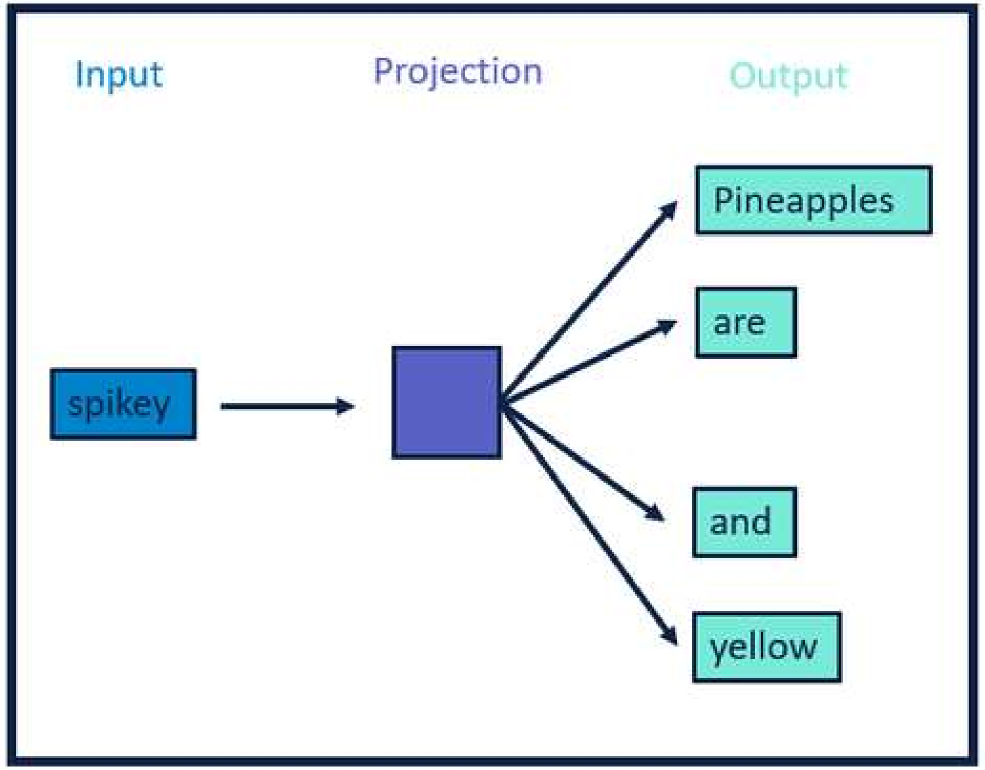
Word2vec

- converts words into **DENSE** numerical vectors (embeddings), making them easier to process by machines.
- The core idea is to capture the meaning of words based on their context in large text datasets
- The model learn word meanings based on the words they commonly appear near.
- Word2Vec transforms words into vectors that capture the meaning based on the context they appear in, allowing machines to process and understand language in a more meaningful way.





word2vec



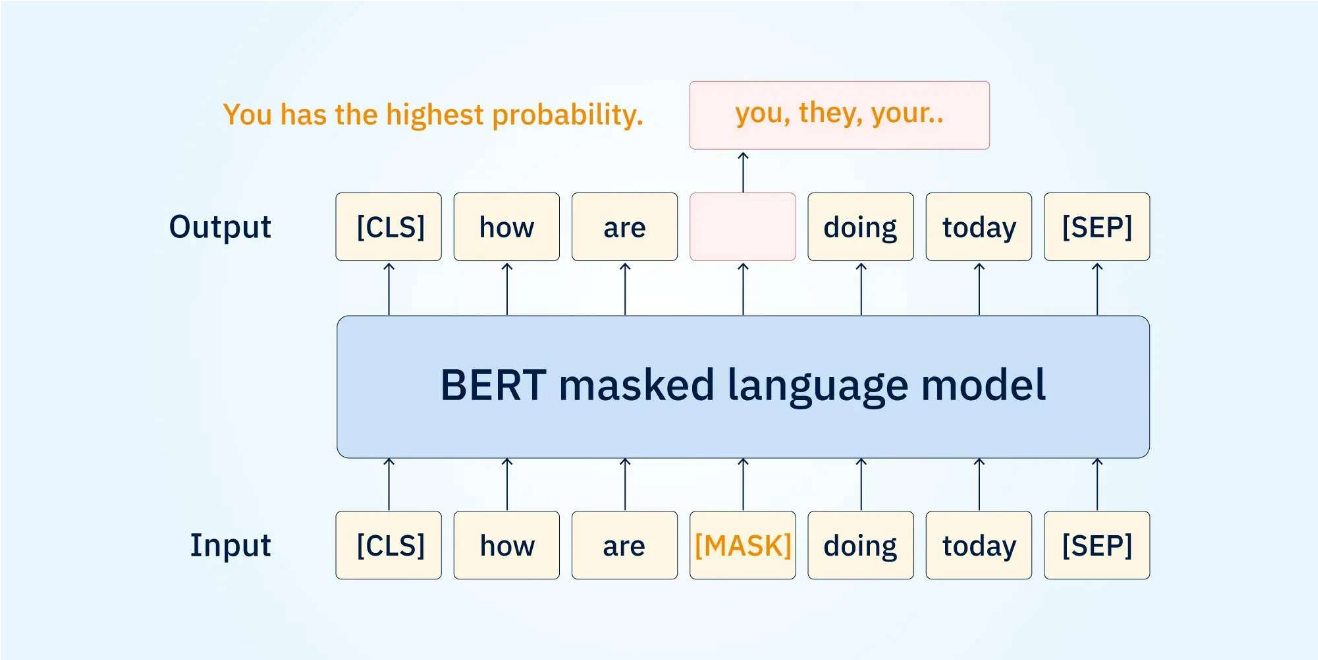
https://community.alteryx.com/t5/Data-Science/Word2vec-for-the-Alteryx-Community/ba-p/305285

BERT

- developed by Google in 2018
- Has learned grammar, context, and tokens from several gigabytes of unstructured data from 2.5 billion words from Wikipedia and another 800 million words from the BookCorpus. It can transform text into a fixed length vectors of size 768
- Contextual Understanding of Words
  - BERT, takes context into account. It looks at the **entire sentence** (both the words before and after a given word) to understand its meaning. This allows BERT to understand words in a **contextual way**

BERT has learnt:

Masked phrases Prediction



Next Sentence Prediction