MONASH University

MONASH
INFORMATION
TECHNOLOGY

Machine Learning- Featurization

GROUP OF EIGHT AUSTRALIA

Machine Learning: Pipeline

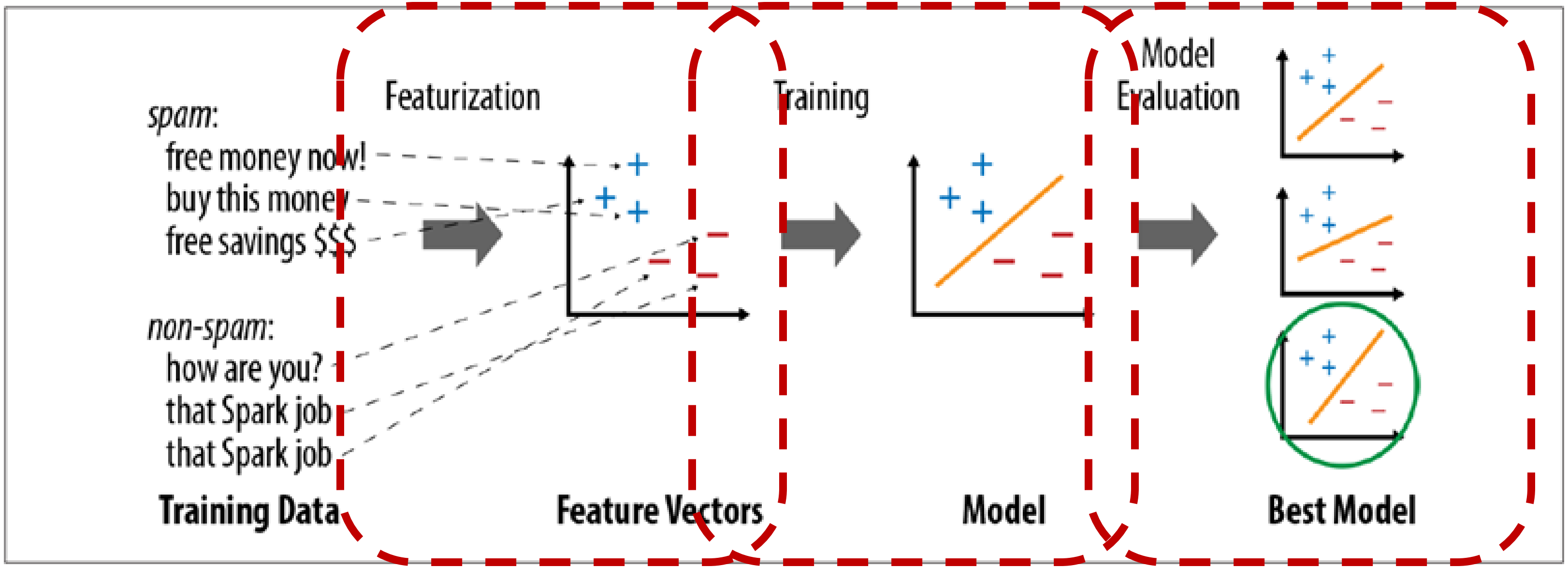


Figure 11-1. Typical steps in a machine learning pipeline

MONASH University

Machine Learning: Pipeline

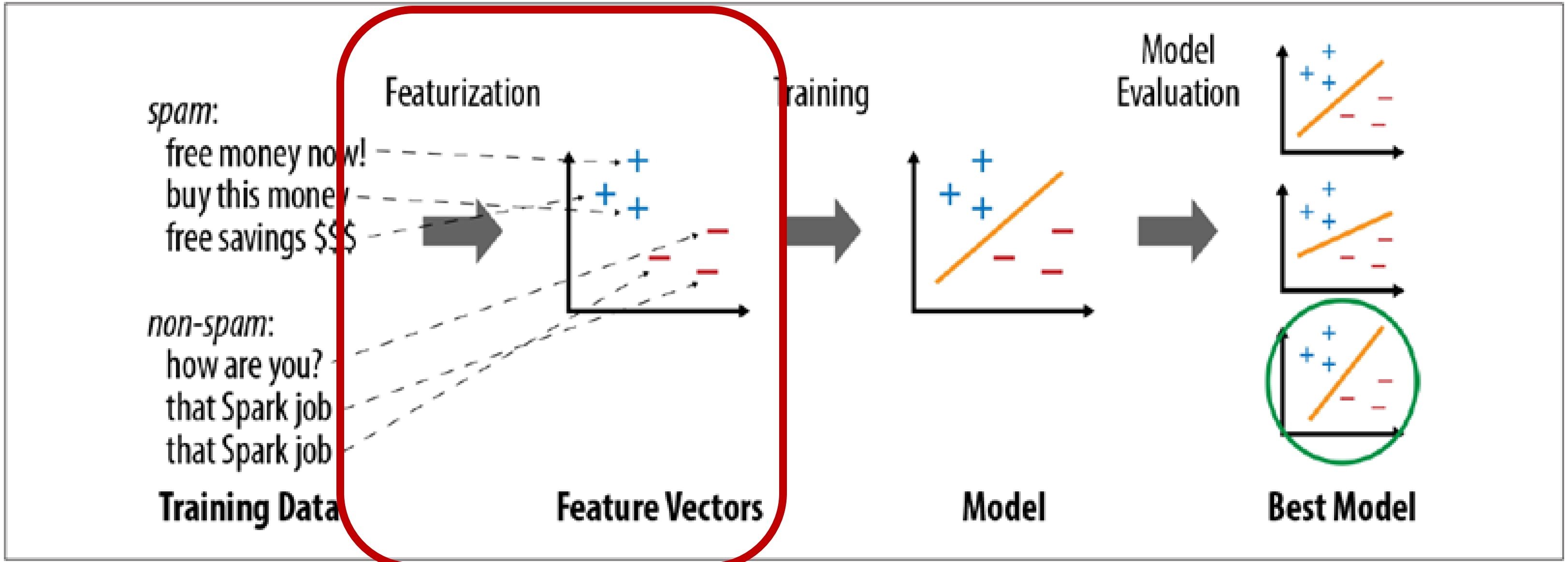


Figure 11-1. Typical steps in a machine learning pipeline

Featurization: Extraction, transformation and selection

- **Extraction**
  - Extracting features from "raw" data
- **Transformation**
  - Scaling, converting, or modifying features
- **Selection**
  - Selecting a subset from a larger set of features

Featurization: Feature Extraction and Transformation

- **Features**
  - Any machine learning algorithm requires some training data. In training data, we have values for all features for all historical records. Consider this simple data set

Height	Weight	Age	Class
165	70	22	Male
160	58	22	Female

Not all features are informative for gender classification

  - We can prepare training data by following two techniques
    - *Feature Extraction–transform raw data into numerical features useable for ML model*
    - *Feature Selection–select a subset of relevant features (e.g., to improve prediction accuracy)*

Featurization: Feature Extraction and Transformation

- **Feature extractors**
  - CountVectorizer
  - TF-IDF
  - Word2Vec
  - FeatureHasher (homework)

Mainly for text processing

Featurization: Feature Extractors

- **Count Vectorizer**
  - Convert a collection of text documents to vectors of token counts.
  - Represent a document with a vector of token/words counts/occurrence
  - During the fitting process, *Count Vectorizer* will build a vocabulary that only considers the top *vocabSize* words ordered by term frequency across the corpus.

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

A corpus – a set of documents

```
 id | texts                           | vector
----|---------------------------------|---------------
 0  | Array("a", "b", "c")            | (3,[0,1,2],[1.0,1.0,1.0])
 1  | Array("a", "b", "b", "c", "a")  | (3,[0,1,2],[2.0,2.0,1.0])
```

id	"a"	"b"	"c"
0	1	1	1
1	2	2	1

Featurization: Feature Extractors

- **Term Frequency–Inverse Document Frequency, or TF-IDF**,
  - A simple way to generate feature vectors from text documents (e.g., web pages).
  - It computes two statistics for each term in each document:
    - ***Term frequency (TF)*** - the number of times a term occurs in a document
    - ***Inverse document frequency (IDF)*** - measures how (in)frequently a term occurs across the whole document corpus.

Featurization: Feature Extractors

- **Term Frequency–Inverse Document Frequency, or TF-IDF**,
  - Measure importance of a term to a document in the corpus
  - Denote a term by ***t,*** a document by ***d,*** and the corpus by ***D*** (collection of documents).
  - Term frequency ***TF(t,d):*** Number of times that term ***t*** appears in document ***d,***

Featurization: Feature Extractors

- **Term Frequency–Inverse Document Frequency, or TF-IDF**,

Suppose that we have term count tables of a corpus consisting of only two documents, as listed on the right.

Calculate TF-IDF for the term "this".

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Featurization: Feature Extractors

- **TF-IDF (Solution)**,

Term frequency ***TF(t,d):*** Number of times that term ***t*** appears in document ***d,***

Calculating TF for "this":

TF ("this", d1) = 1/5 = 0.2
TF ("this", d2) = 1/7 = 0.14 (Approx.)

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Limitation of ***TF(t,d):*** over-emphasize terms that appear very often but carry little information about the document, e.g., "a", "the" and "of"

Featurization: Feature Extractors

- **Term Frequency–Inverse Document Frequency, or TF-IDF**,
  - Inverse document frequency ***IDF(t,D)***: Numerical measure of how much information a term provides:

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

- ❑ ***|D|*** is the total number of documents in the corpus.
- ❑ Document frequency ***DF(t,D)*** is the number of documents that contains term ***t.***
- ❑ If ***DF(t,D)*** = ***|D|*** (all documents contain term ***t***), ***IDF(t,D)=0***

Featurization: Feature Extractors

- **TF-IDF (Solution)**,

Low values of IDF ➔ A term appears very often across corpus, and it does not carry special information about a document

Calculating IDF for "this":
|D| = 2
DF ("this", D) = 2

IDF ("this", D) = log (3/ 3) = 0

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

where **|D|** is the total number of documents in the corpus.
DF(t,D) is the number of documents that contains term ***t***

Featurization: Feature Extractors

- **Term Frequency–Inverse Document Frequency, or TF-IDF**,
  - The product of these values, TF × IDF, shows how relevant a term is to a specific document (i.e., if it is common in that document but rare in the whole corpus).
  - The TF-IDF measure is simply the product of TF and IDF:

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D).$$

Featurization: Feature Extractors

- **TF-IDF (Solution),**

Calculating TF-IDF for "this":

TF-IDF ("this", d1, D) = 0.2 * 0 = 0

TF-IDF ("this", d2, D) = 0.14 * 0 = 0

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D).$$

The term "this" is not of importance to both the documents in the corpus.

Featurization: Feature Extractors

- **Exercise: Calculate TF-IDF for the term "example".**

Suppose that we have term count tables of a corpus consisting of only two documents, as shown below.

Calculate TF-IDF for the term "example".

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Featurization: Feature Extractors

- **TF-IDF (Solution),**

Calculating TF-IDF for “example”:

TF (“example”, d1) = 0/5 = 0
TF (“example”, d2) = 3/7 = 0.429

IDF(“example”, D) = log(3/2) = 0.584
(using log base 2)

TF-IDF (“example”, d1, D) = 0 * 0.584 = 0
TF-IDF (“example”, d2, D) = 0.429 * 0.584 = 0.250

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

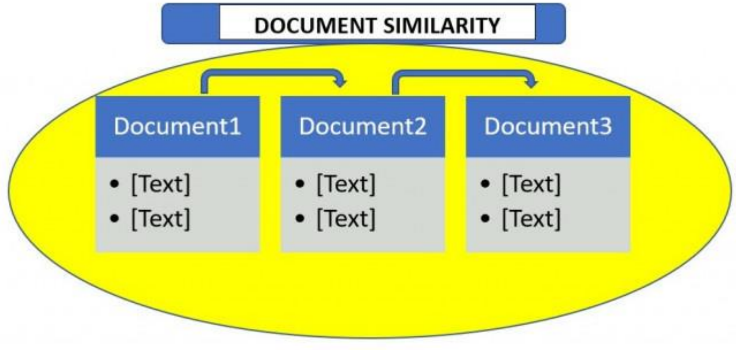
Term	Term Count
this	1
is	1
another	2
example	3

$$IDF(t,D) = \log \frac{|D|+1}{DF(t,D)+1},$$

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D).$$

Featurization: Feature Extractors

- **Word2Vec**
  - maps each word to a unique fixed-size vector.
  - transforms each document into a vector using the average of all words in the document.
  - this vector can then be used as features for prediction, **document similarity calculations**, etc.



Featurization: Extraction, transformation and selection

- **Extraction**
  - Extracting features from “raw” data
- **Transformation**
  - Scaling, converting, or modifying features
- **Selection**
  - Selecting a subset from a larger set of features

Featurization: Feature Extraction and Transformation

- **Feature Transformers**
  - Tokenization
  - Stop Words Remover
  - String Indexing
  - One Hot Encoding
  - Vector Assembler

Featurization: Feature Transformers

- **Tokenization**
  – It is the process of taking text (such as a sentence) and breaking it into individual terms (usually words).

Text
"The cat sat on the mat."
↓
Tokens
"the", "cat", "sat", "on", "the", "mat", "."

Monash University

Featurization: Feature Transformers

- **Stop Words** are words which should be excluded from the input, typically because the words appear frequently and don't carry as much meaning.

Some words contain more information than others

stopwords — [the, in, for, you, will, have, be]

Quiz: How many words will be removed when we remove stopwords from "Hi Katie the machine learning class will be great best Sebastian"

3

Featurization: Feature Transformers

- **Stop Words Remover**
  - Takes as input a sequence of strings (e.g. the output of a Tokenizer)
  - Drops all the stop words from the input sequences.

```
 id | raw                          | filtered
----|------------------------------|---------------------
 0  | [I, saw, the, red, balloon]  |  [saw, red, balloon]
 1  | [Mary, had, a, little, lamb]|[Mary, little, lamb]
```

Featurization: Feature Transformers

- **String Indexing**
  - Encoding a string column of labels to a column of **label indices** (in numerical/floating point numbers), understandable by ML algorithms

```
 id | category | categoryIndex
----|----------|---------------
 0  | a        | 0.0
 1  | b        | 2.0
 2  | c        | 1.0
 3  | a        | 0.0
 4  | a        | 0.0
 5  | c        | 1.0
```

Featurization: Feature Transformers

- **One Hot Encoding**
  - Maps a categorical feature represented as a label index to a binary vector.
  - A single one-value indicates the presence of a specific feature value from among the set of all feature values.
  - For string type input data, it is common to encode categorical features using String Indexing first.

Example, three category labels {a,b,c}

$$'a' - \begin{bmatrix}1\\0\\0\end{bmatrix} \quad 'b' - \begin{bmatrix}0\\1\\0\end{bmatrix} \quad 'c' - \begin{bmatrix}0\\0\\1\end{bmatrix}$$

In spark, use *N*-1 dimensional binary vector (*N* is the number of categories)

$$'a'(index\ 0) - \begin{bmatrix}1\\0\end{bmatrix} \quad 'b'(index\ 1) - \begin{bmatrix}0\\1\end{bmatrix} \quad 'c'(index\ 2) - \begin{bmatrix}0\\0\end{bmatrix}$$

Featurization: Feature Transformers

- **Why One Hot Encoding?**
  - Example: Let's say we have 3 data instances with attributes of Preferred Programming Language and OS of Choice.

Preferred Programming Language	OS of Choice
Javascript	OSX
Python	Linux
Scala	OSX

Featurization: Feature Transformers

- **Why One Hot Encoding?**

Preferred Programming Language	OS of Choice
Javascript	OSX
Python	Linux
Scala	OSX

String Indexing

Preferred Programming Language	OS of Choice
0	0
1	1
2	0

Featurization: Feature Transformers

- **Why can't we STOP here?**
- **The Problem Of Ordinality**
- Machine learning algorithms treat the ordinality of numbers in an attribute with some significance: *a higher number "must be better" than a lower number*.

String Indexing

Preferred Programming Language	OS of Choice
0	0
1	1
2	0

MONASH University

Featurization: Feature Transformers

Preferred Programming Language	OS of Choice
Javascript	OSX
Python	Linux
Scala	OSX

- **One Hot Encoding**

Javascript	Python	Scala	OSX	Linux
1	0	0	1	0
0	1	0	0	1
0	0	1	1	0

Featurization: Feature Transformers

- **Why One Hot Encoding?**
  - For categorical variables when there is no ordinal relationship, the string indexing is not enough...
  - Using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results.
  - A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

Featurization: Extraction, transformation and selection

- **Extraction**
  - Extracting features from “raw” data
- **Transformation**
  - Scaling, converting, or modifying features
- **Selection**
  - Selecting a subset from a larger set of features

Featurization: Feature Selectors

- **Feature selection**
  - This process tries to get most important features that are contributing to decide the label.
  - **Vector Slicer**
    - It takes a feature vector and outputs a new feature vector with a sub-array of the original features.
    - It is useful for extracting features from a vector column.

```
userFeatures      | features
------------------|-------------
[0.0, 10.0, 0.5] | [10.0, 0.5]
```