Text Classification

Machine Learning Bootcamp

- Section Goals
  - Understand Machine Learning Basics
  - Understand Classification Metrics
  - Understand Text Feature Extraction
  - Familiarize ourselves with Scikit-Learn and Python to perform text classification on real data sets.

Machine Learning Bootcamp

- Keep in mind, we start with quite a few “theory” lectures, we won’t code anything until we have a solid understanding of Machine Learning, Classification, and Text Feature Extraction concepts.
- Let’s get started!

Machine Learning Overview

Machine Learning

- Before we dive into Text Classification, let's work on understanding the general machine learning process we will be using.
- The specific case of machine learning we will be conducting is known as **supervised learning**

Machine Learning

- We will keep the mathematics behind the machine learning algorithms light.

Machine Learning

- A great textbook on general machine learning is **Introduction to Statistical Learning** by Gareth James as a companion book.
- It’s freely available online. Simply google search the title of the book.

What is Machine Learning?

- Machine learning is a method of data analysis that automates analytical model building.

- Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.

What is it used for?

- Fraud detection.
- Web search results.
- Real-time ads on web pages
- Credit scoring and next-best offers.
- Prediction of equipment failures.
- New pricing models.
- Network intrusion detection.
- Recommendation Engines
- Customer Segmentation
- Text Sentiment Analysis
- Predicting Customer Churn
- Pattern and image recognition.
- Email spam filtering.

Supervised Learning

- **Supervised learning** algorithms are trained using **labeled** examples, such as an input where the desired output is known.
- For example, a segment of text could have a category label, such as:
  - **Spam** vs. **Legitimate** Email
  - **Positive** vs. **Negative** Movie Review

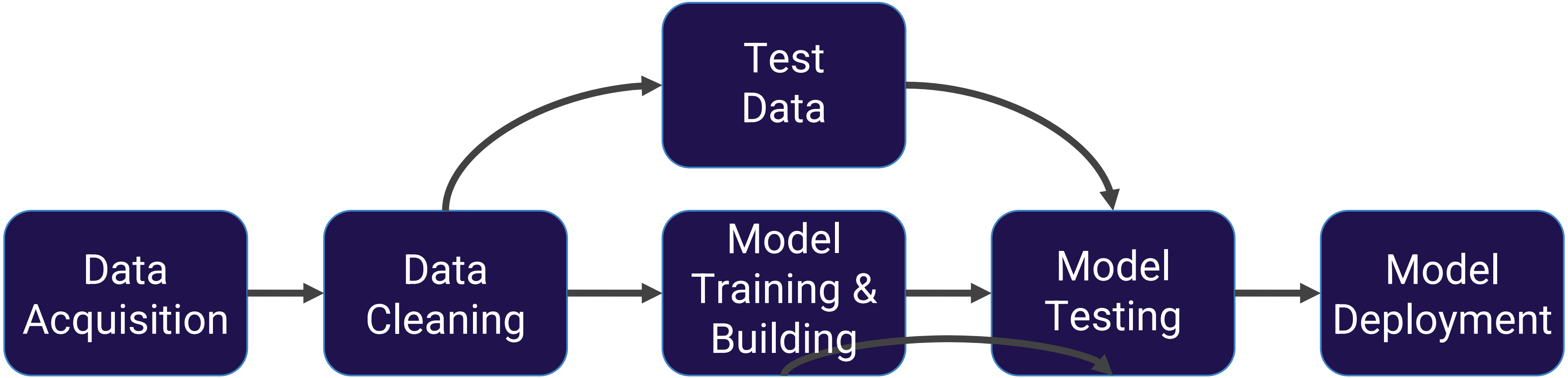
Supervised Learning

- The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors.
- It then modifies the model accordingly.

Supervised Learning

- Supervised learning is commonly used in applications where historical data predicts likely future events.

Machine Learning Process



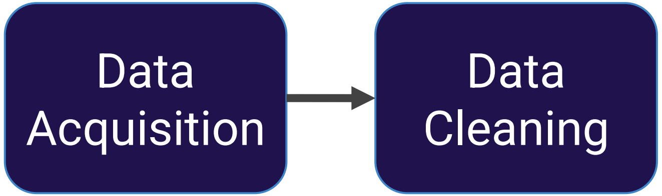
Machine Learning Process

- Get your data! Customers, Sensors, etc...

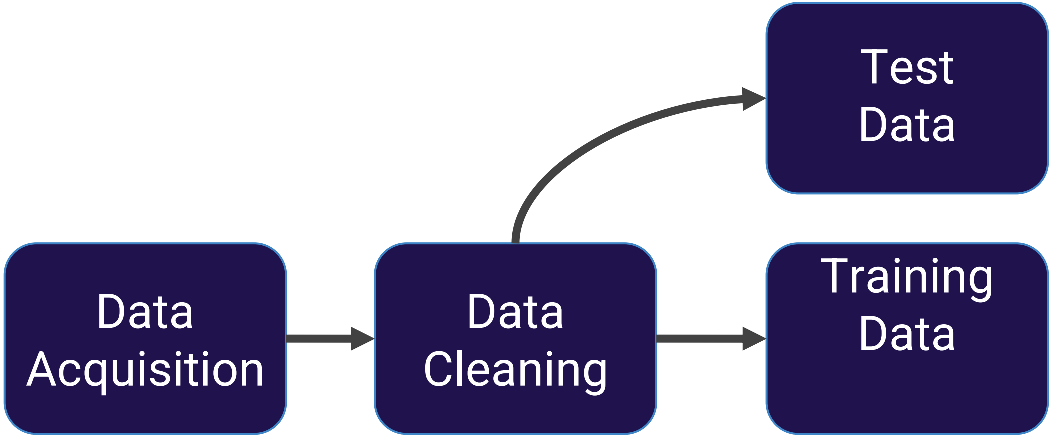
Data Acquisition

Machine Learning Process

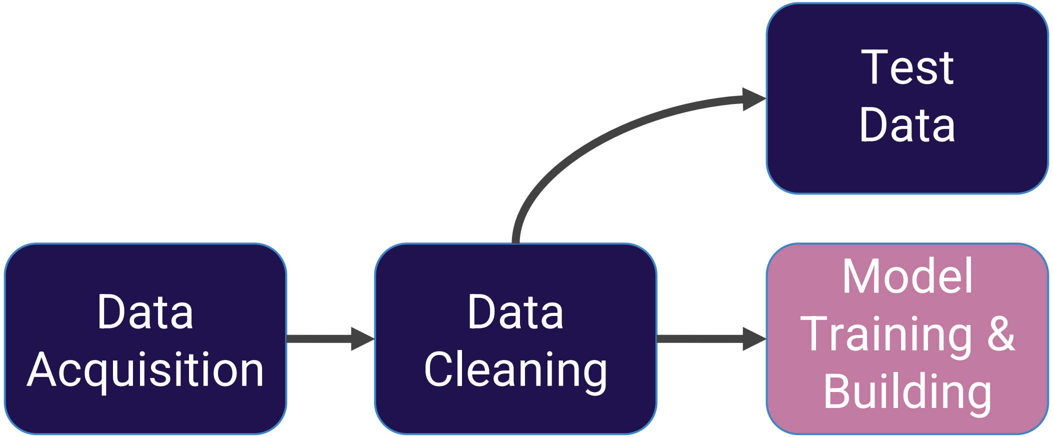
- Clean and format your data (using SciKit Learn and Vectorization)



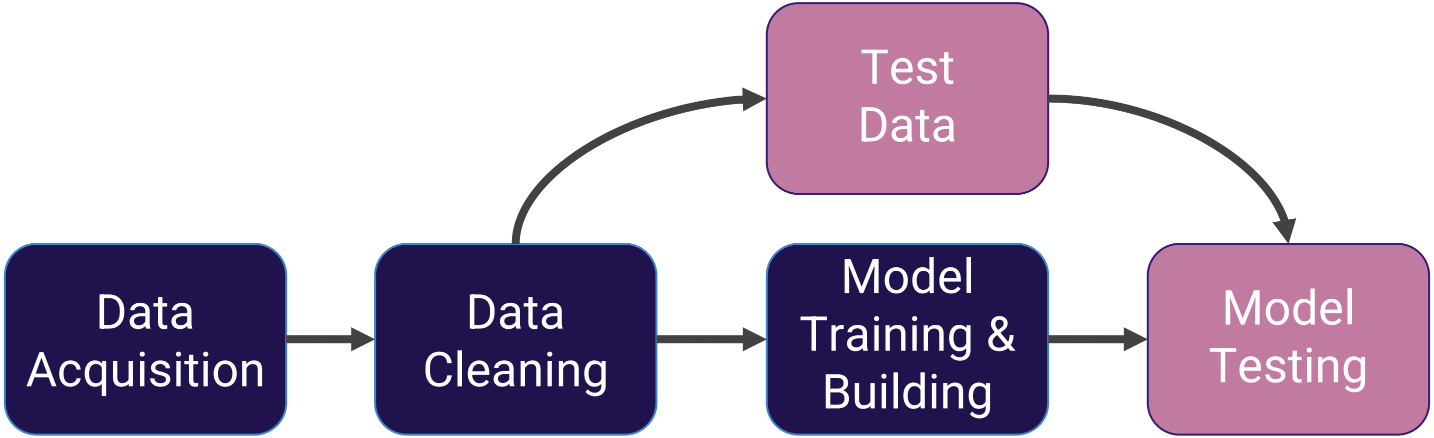
Machine Learning Process



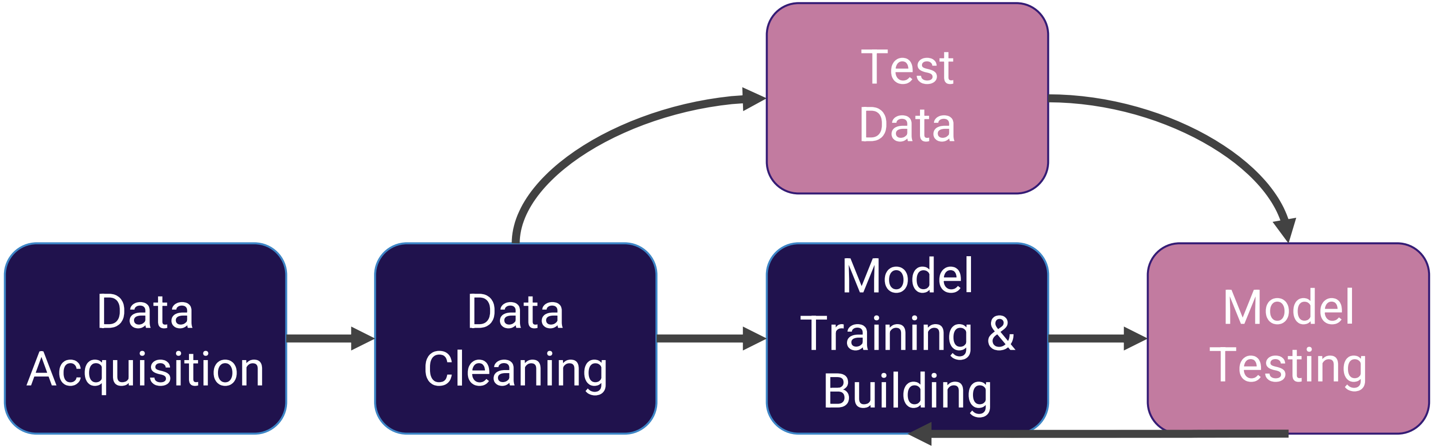
Machine Learning Process



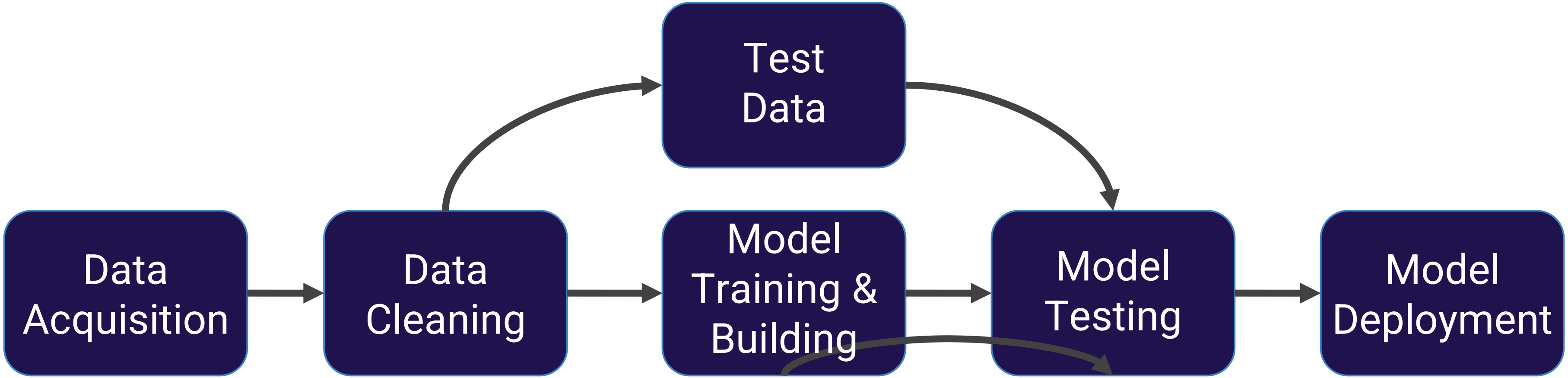
Machine Learning Process



Machine Learning Process



Machine Learning Process



Supervised Learning

- Text classification and recognition is a very common and widely applicable use of machine learning.
- Later on we will learn about the SciKit-Learn Library in order to use Python to conduct machine learning text classification!

Model Evaluation

- Let’s take a moment to focus on the train/test split that occurred and learn a few terms.
- Let’s imagine a full data set of Ham vs Spam text messages.

Model Evaluation

- Let's imagine a full data set of Ham vs Spam text messages.

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

Model Evaluation

- Before the split, we have labels and features

Y Label

X Features

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

Model Evaluation

- We call these Y Labels and X Features

Y Label

X Features

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

Model Evaluation

- Before we fit the model, we split the data!

Y Label

X Features

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

Model Evaluation

- The data is split

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...

spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

Model Evaluation

- Test and Train Data Sets

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...

TEST

spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

TRAIN

Model Evaluation

- Before we fit the model, we split the data!

Y TEST

X TEST

TEST

label	message
ham	Go until jurong point, crazy.. Available only ...
ham	Ok lar... Joking wif u oni...

Y TRAIN

X TRAIN

TRAIN

spam	Free entry in 2 a wkly comp to win FA Cup fina...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives aro...

Model Evaluation

- Notice how after a train test split we always end up with 4 components:
  - X_train
  - X_test
  - Y_train
  - Y_test

Model Evaluation

- These 4 components are simply the result of the train/test split groups being separated between features and labels.
- Let’s continue to understand classification process in more detail and metrics to evaluate it!

Classification Metrics

Model Evaluation

- We just learned that after our machine learning process is complete, we will use performance metrics to evaluate how our model did.
- Let's discuss classification metrics in more detail!

Model Evaluation

- The key classification metrics we need to understand are:
  - Accuracy
  - Recall
  - Precision
  - F1-Score

Model Evaluation

But first, we should understand the reasoning behind these metrics and how they will actually work in the real world!

Model Evaluation

- Typically in any classification task your model can only achieve two results:
  - Either your model was **correct** in its prediction.
  - Or your model was **incorrect** in its prediction.

Model Evaluation

- Fortunately incorrect vs correct expands to situations where you have multiple classes.
- For the purposes of explaining the metrics, let’s imagine a **binary classification** situation, where we only have two available classes.

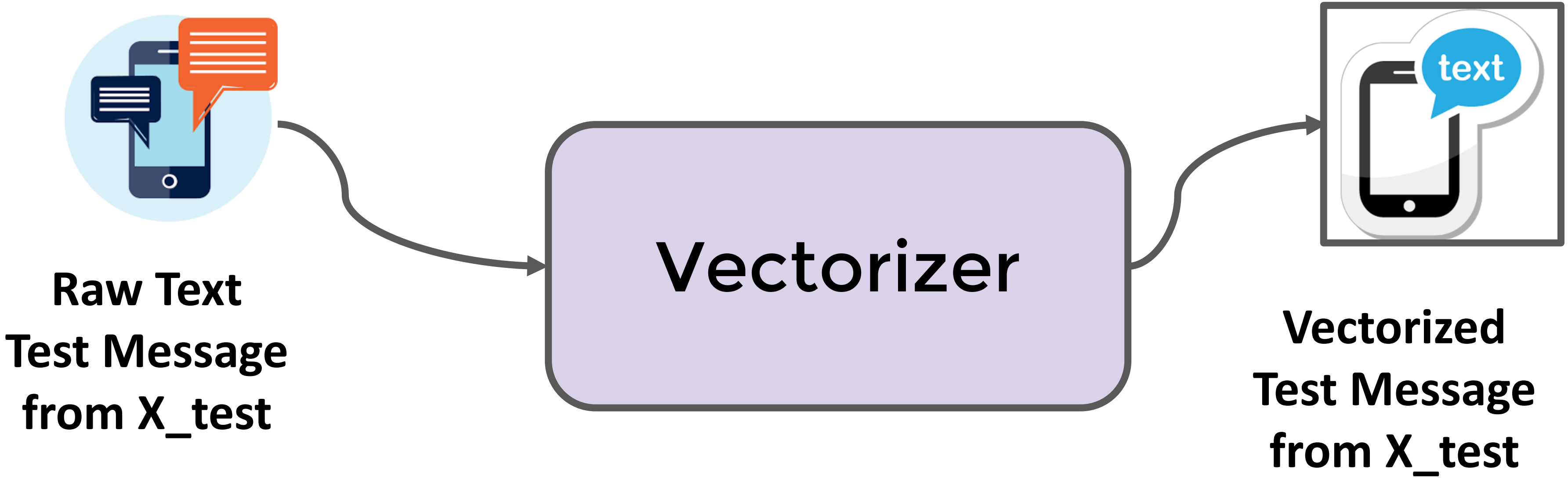
Model Evaluation

- In our example, we will attempt to predict if a text is **Spam** or **Ham (legitimate)**.
- Since this is supervised learning, we will first **fit/train** a model on **training data**, then **test** the model on **testing data**.
- Once we have the model's predictions from the **X_test** data, we compare it to the **true y values** (the correct labels).

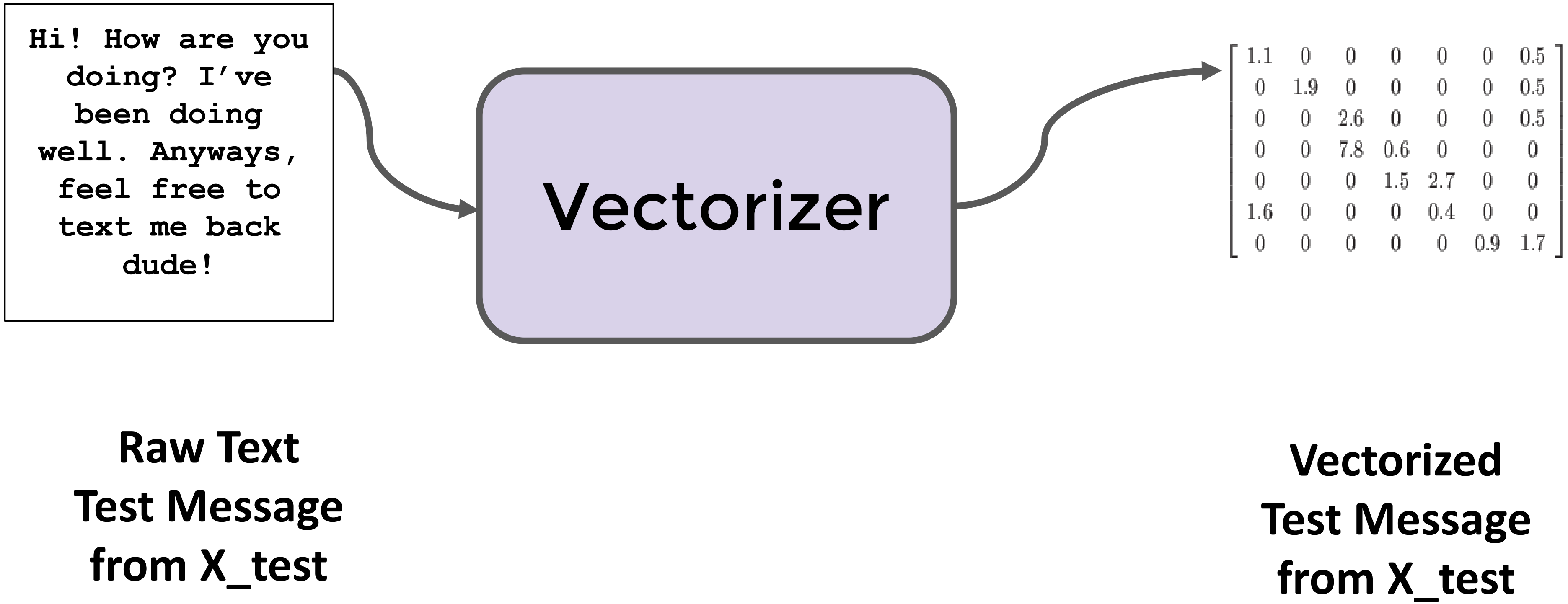
Model Evaluation

- Keep in mind, there will be a few steps to convert the raw text into a format that the machine learning model can understand.
- We will discuss these methods in much more detail later on!

Model Evaluation



Model Evaluation



Model Evaluation

- We set up this vectorization in a pipeline and there are many ways of transforming the raw text into numerical information.
- For now, let’s focus on the classification process and assume there is some underlying vectorization.

Model Evaluation

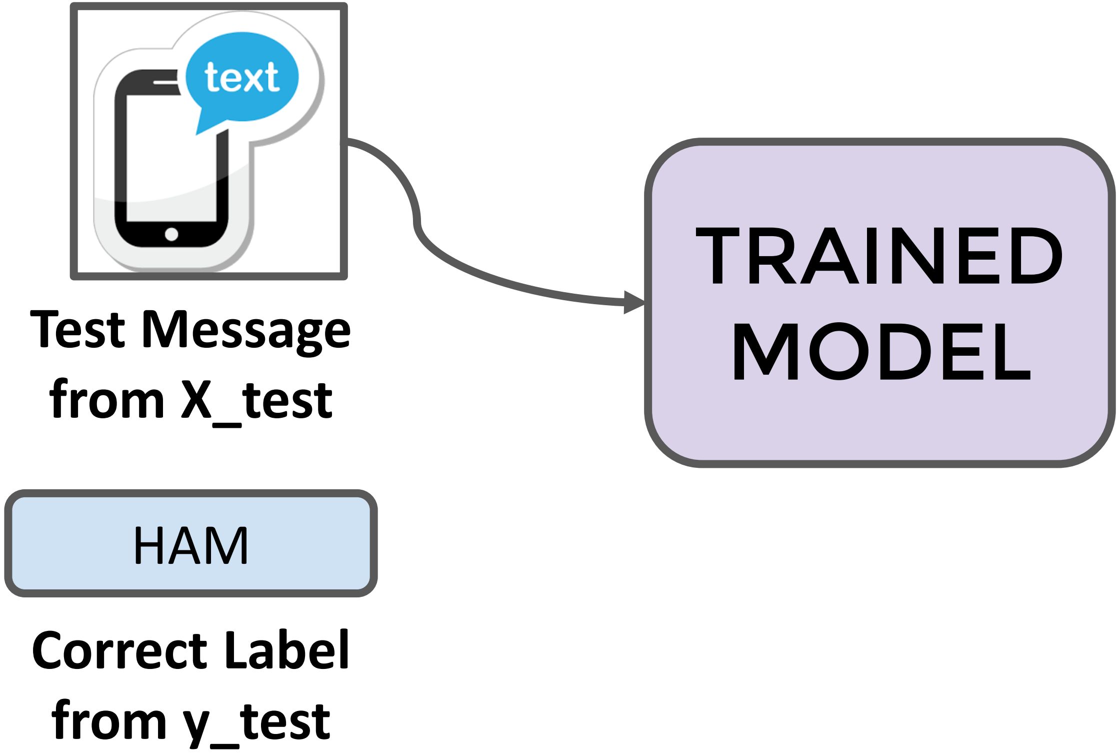
TRAINED MODEL

Model Evaluation

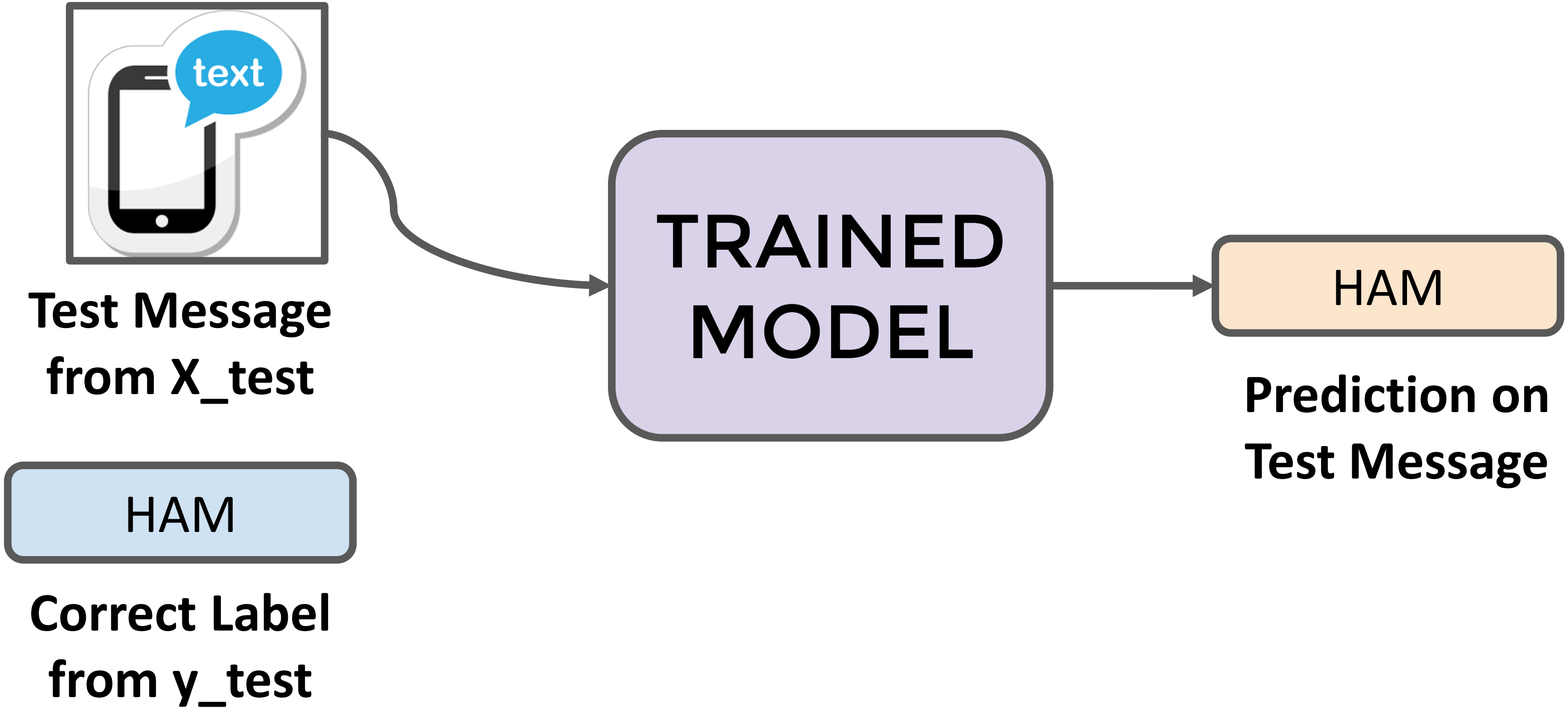
Vectorized Test Message from X_test

TRAINED MODEL

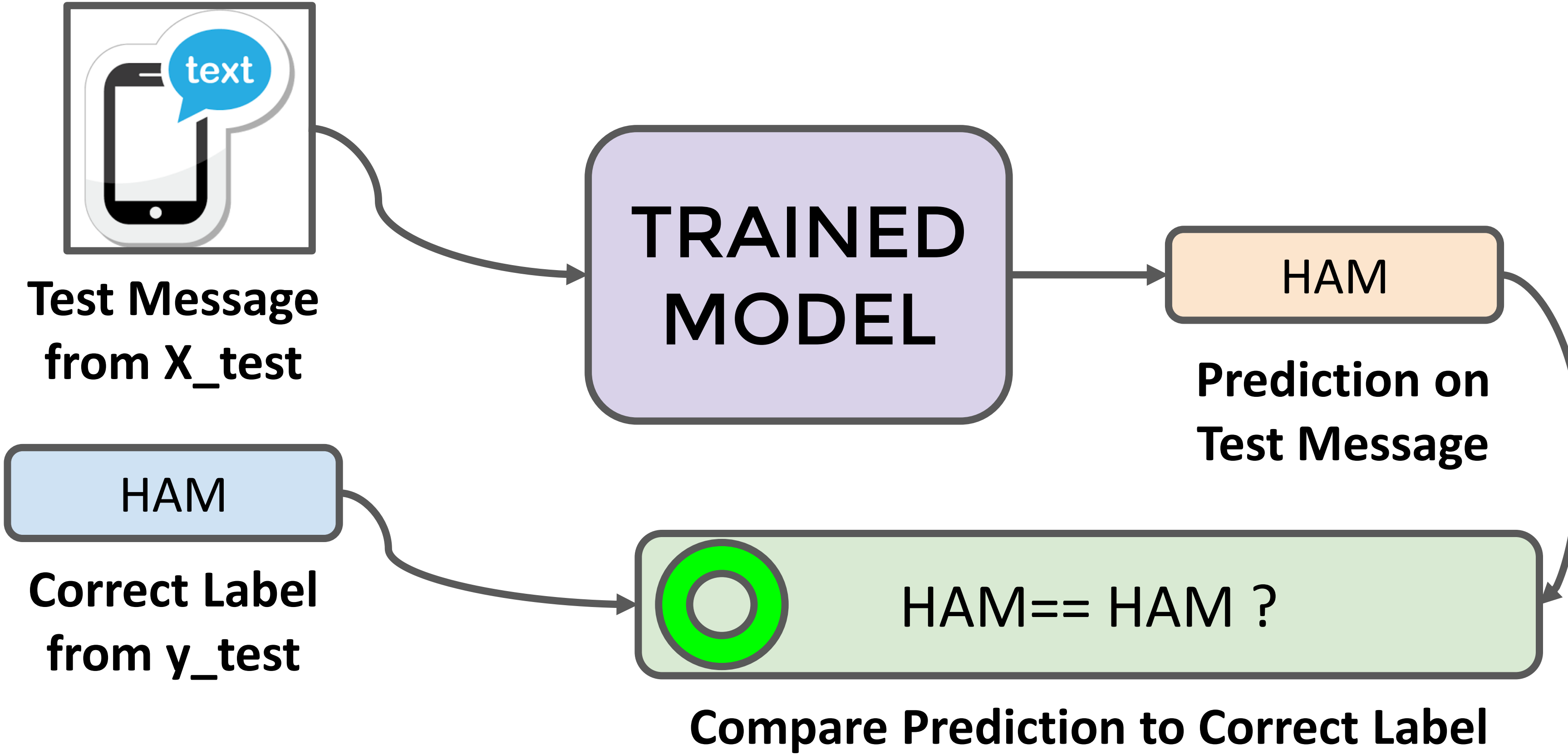
Model Evaluation



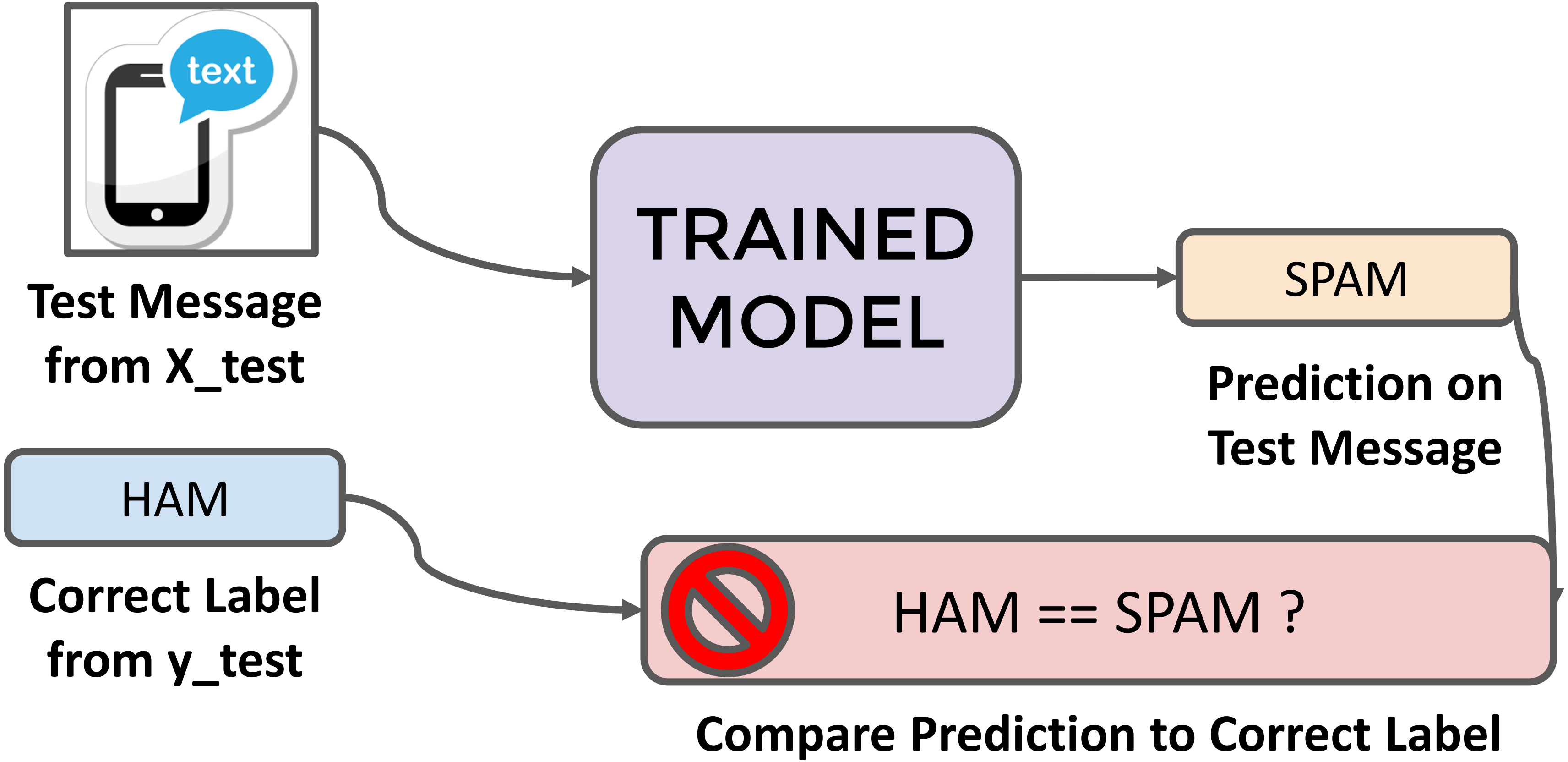
Model Evaluation



Model Evaluation



Model Evaluation



Model Evaluation

- We repeat this process for all the text messages in our X test data.
- At the end we will have a count of correct matches and a count of incorrect matches.
- The key realization we need to make, is that **in the real world, not all incorrect or correct matches hold equal value !**

Model Evaluation

- Also in the real world, a single metric won't tell the complete story !
- To understand all of this, let's bring back the 4 metrics we mentioned and see how they are calculated.
- We could organize our predicted values compared to the real values in a **confusion matrix.**

Model Evaluation

- Accuracy
    - Accuracy in classification problems is the **number of correct predictions** made by the model divided by the **total number of predictions.**

Model Evaluation

- Accuracy
  - For example, if the X_test set was 100 messages and our model **correctly** predicted 80 messages, then we have **80/100.**
  - **0.8** or **80% accuracy.**

Model Evaluation

- Accuracy
  - Accuracy is useful when target classes are well balanced.
  - In our example, we would have roughly the same amount of spam messages as we have ham messages.

Model Evaluation

- Accuracy
  - Accuracy is **not** a good choice with **unbalanced** classes!
  - Imagine we had 99 legitimate ham messages and 1 spam text message.
  - If our model was simply a line that always predicted **HAM** we would get 99% accuracy!

Model Evaluation

- Accuracy
  - Accuracy is **not** a good choice with **unbalanced** classes!
  - Imagine we had 99 legitimate ham messages and 1 spam text message.
  - In this situation we'll want to understand **recall** and **precision!**

Model Evaluation

- Let's quickly go over some formal definitions of Precision, Recall, and F1-Score (a combination of Precision and Recall).

Model Evaluation

- Recall
  - Ability of a model to find <u>**all**</u> the relevant cases within a dataset.
  - The precise definition of recall is the **number of true positives divided by (the number of true positives plus the number of false negatives).**
  - **99 / (99+0) = 1 for Ham**
  - **0 / (0+0) = 0 for Spam**

Model Evaluation

- Precision
  - Ability of a classification model to identify <u>only</u> the relevant data points.
  - Precision is defined as the number of **true positives divided by (the number of true positives plus the number of false positives).**
  - **99/(99+1) = 0.99 for Ham**
  - **0 / (0+ 1) = 0 for Spam**

Model Evaluation

- Recall and Precision
  - Often you have a trade-off between Recall and Precision.
  - While recall expresses the ability to find all relevant instances in a dataset, precision expresses the proportion of the data points our model says was relevant actually were relevant.

Model Evaluation

- F1-Score
  - In cases where we want to find an optimal blend of precision and recall we can combine the two metrics using what is called the F1 score.

Model Evaluation

- F1-Score
  - The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Model Evaluation

- F1-Score
  - We use the harmonic mean instead of a simple average because it punishes extreme values.
  - A classifier with a precision of 1.0 and a recall of 0.0 has a simple average of 0.5 but an F1 score of 0.

Model Evaluation

- Precision and Recall typically make more sense in the context of a confusion matrix.
- In the next lecture we will explore the confusion matrix!

Confusion Matrix

Model Evaluation

- We mentioned a way to view various metrics of classification is the **confusion matrix.**
- Let’s explore the basics of the confusion matrix.

Model Evaluation

- In a classification problem, during the testing phase you will have Two Categories:
  - True Condition
  - Predicted Condition

Model Evaluation

- In a classification problem, during the testing phase you will have Two Categories:
  - True Condition
    - A text message is SPAM
  - Predicted Condition
    - ML Model predicted SPAM

Model Evaluation

- In a classification problem, during the testing phase you will have Two Categories:
  - True Condition
    - A text message is SPAM
  - Predicted Condition
    - ML Model predicted HAM

Model Evaluation

- This means if you have two possible classes you should have 4 separate groups at the end of testing:
- Correctly classified to Class 1: TRUE HAM
- Correctly classified to Class 2: TRUE SPAM
- **Incorrectly** classified to Class 1: FALSE HAM
- **Incorrectly** classified to Class 2: FALSE SPAM

Confusion Matrix

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Confusion Matrix

		PREDICTED CONDITION	
	ALL TEXTS	PREDICTED HAM	PREDICTED SPAM
REAL CONDITION	REAL CONDITION HAM	TRUE POSITIVE	FALSE NEGATIVE
	REAL CONDITION SPAM	FALSE POSITIVE	TRUE NEGATIVE

Confusion Matrix

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition HAM	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Confusion Matrix

		predicted condition		
	total population	prediction positive	prediction negative	Prevalence $= \frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection $= \frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**	False Positive Rate (FPR), Fall-out, Probability of False Alarm $= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$
	Accuracy $= \frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision $= \frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) $= \frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$
		False Discovery Rate (FDR) $= \frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$

Model Evaluation

- The main point to remember with the confusion matrix and the various calculated metrics is that they are all fundamentally ways of comparing the predicted values versus the true values.
- What constitutes “good” metrics, will really depend on the specific situation!

Model Evaluation

- We can use a confusion matrix to evaluate our model.
- For example, imagine testing for disease.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Example: Test for presence of disease
NO = negative test = False = 0
YES = positive test = True = 1

Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Basic Terminology:

- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy:

- Overall, how often is it **correct**?
- (TP + TN) / total = 150/165 = 0.91

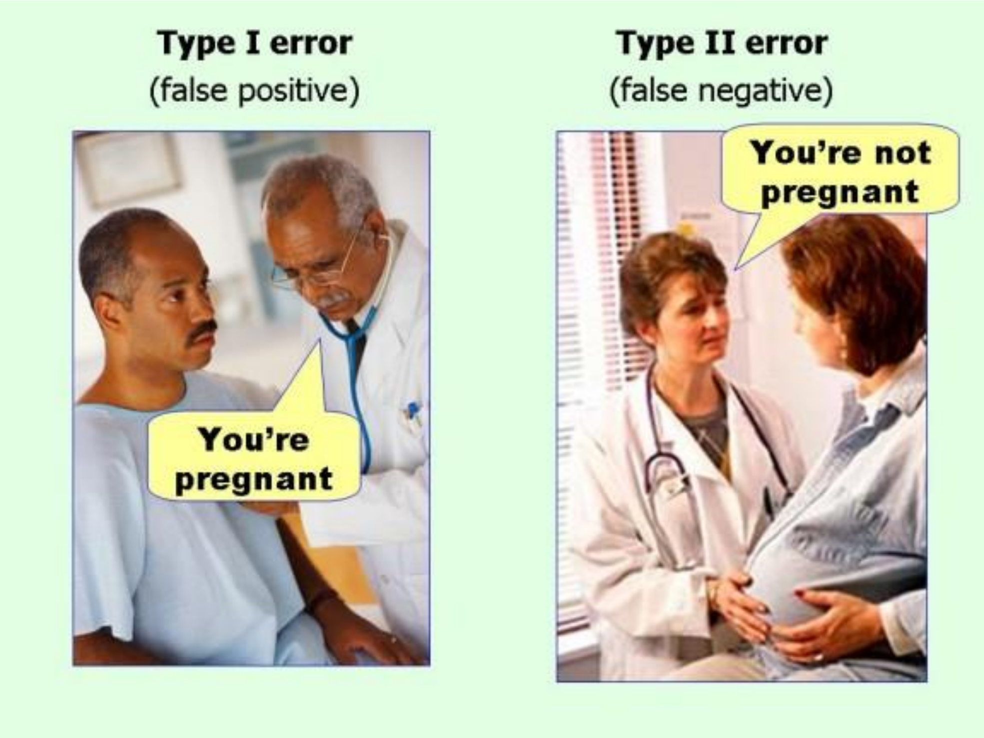
Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Misclassification Rate (Error Rate):

- Overall, how often is it **wrong**?
- (FP + FN) / total = 15/165 = 0.09

Confusion Matrix



Model Evaluation

- Still confused on the confusion matrix?
- No problem! Check out the Wikipedia page for it, it has a really good diagram with all the formulas for all the metrics.
- Throughout the training, we'll usually just print out metrics (e.g. accuracy).

Scikit-Learn Primer

Scikit Learn

We will be using the **Scikit Learn** package.

It's the most popular machine learning package for Python and has a lot of algorithms built-in!

Scikit Learn

You may need to install it using:

conda install scikit-learn

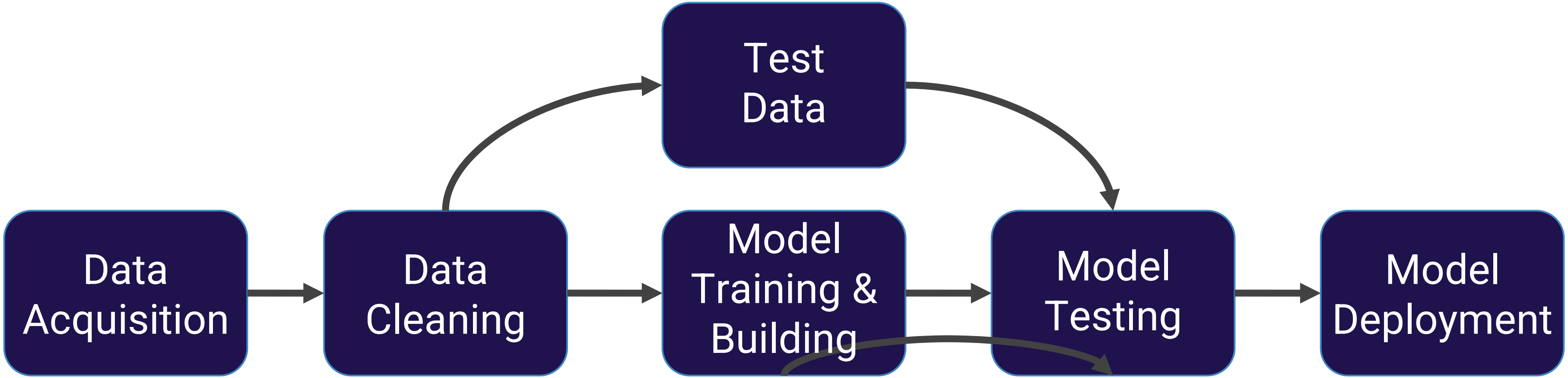
or

pip install scikit-learn

Scikit Learn

- Let’s talk about the basic structure of how to use Scikit Learn!
- First, a quick review of the machine learning process.

Machine Learning Process



Scikit Learn

- Now let’s go over an example of the process to use SciKit Learn.
- Don’t worry about memorizing any of this, we’ll get plenty of practice and review when we actually start coding in subsequent lectures!

Scikit Learn

- Every algorithm is exposed in scikit-learn via an "Estimator"
- First you'll import the model, the general form is:

```
from sklearn.family import Model
```

For example:

```
from sklearn.linear_model import LinearRegression
```

Scikit Learn

Estimator parameters: All the parameters of an estimator can be set when it is instantiated, and have suitable default values.

You can use Shift+tab in jupyter to check the possible parameters.

Scikit Learn

For example:

```
model = LinearRegression(normalize=True)
print(model)
```

LinearRegression(copy_X=True, fit_intercept=True, normalize=True)

Scikit Learn

Once you have your model created with your parameters, it is time to fit your model on some data!

But remember, we should split this data into a training set and a test set.

Scikit Learn

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]
```

Scikit Learn

```
>>> X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3)
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

Scikit Learn

Now that we have split the data, we can train/fit our model on the training data.

This is done through the model.fit() method:

```
model.fit(X_train,y_train)
```

Scikit Learn

- Now the model has been fit and trained on the training data.
- The model is ready to predict labels or values on the test set!

Scikit Learn

We get predicted values using the predict method:

```
predictions = model.predict(X_test)
```

Scikit Learn

We can then evaluate our model by comparing our predictions to the correct values.
The evaluation method depends on what sort of machine learning algorithm we are using (e.g. Regression,Classification, Clustering, etc.)

Scikit-Learn Primer

Coding Lecture

Feature Extraction From Text

Part One

Natural Language Processing Bootcamp

- Most classic machine learning algorithms can’t take in raw text.
- Instead we need to perform a feature “extraction” from the raw text in order to pass numerical features to the machine learning algorithm.

Natural Language Processing Bootcamp

- For example, we could count the occurence of each word to map text to a number.
- Let's discuss Counter Vectorization along with Term-Frequency and Inverse Document Frequency.

Natural Language Processing Bootcamp

- Count Vectorization

```python
messages = ["Hey, lets go to the game today!",
            "Call your sister.",
            "Want to go walk your dogs?"]
```

Natural Language Processing Bootcamp

- Count Vectorization

```python
messages = ["Hey, lets go to the game today!",
            "Call your sister.",
            "Want to go walk your dogs?"]
```

```python
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
```

- Count Vectorization

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
```

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

- ## Count Vectorization

vect.fit_transform(messages)

```
vect.get_feature_names()
```

```
['call',
 'dogs',
 'game',
 'go',
 'hey',
 'lets',
 'sister',
 'the',
 'to',
 'today',
 'walk',
 'want',
 'your']
```

Natural Language Processing Bootcamp

- Document Term Matrix (DTM)

call	dogs	game	go	hey	lets	sister	the	to	today	walk	want	your
0	0	1	1	1	1	0	1	1	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	0	1	0	1	1	1

Natural Language Processing Bootcamp

- An alternative to CountVectorizer is something called TfidfVectorizer. It also creates a document term matrix from our messages.
- However, instead of filling the DTM with token counts it calculates term frequency-inverse document frequency value for each word(TF-IDF).

Natural Language Processing Bootcamp

- Term frequency **tf(t,d)**: is the raw count of a term in a document, i.e. the number of times that term t occurs in document d.

Natural Language Processing Bootcamp

- However, Term Frequency alone isn’t enough for a thorough feature analysis of the text!
- Let’s imagine very common terms, like “a” or “the”...

Natural Language Processing Bootcamp

- Because the term "the" is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "red" and "dogs".

Natural Language Processing Bootcamp

- An inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Natural Language Processing Bootcamp

- It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient)

TF-IDF

- **TF-IDF = term frequency * (1 / document frequency)**
- **TF-IDF = term frequency * inverse document freq**

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

total number of documents in the corpus

number of documents where the term appears

- Fortunately Scikit-learn can calculate all these terms for us through the use of its API.
- Notice how similar the syntax is to our previous use of ML models in Scikit-Learn!

Natural Language Processing Bootcamp

```
from sklearn.feature_extraction.text import TfidfVectorizer

vect = TfidfVectorizer()
dtm = vect.fit_transform(messages)
```

call	dogs	game	go	hey	lets	sister	the	to	today	walk	want	your
0.000	0.00	0.403	0.307	0.403	0.403	0.000	0.403	0.307	0.403	0.00	0.00	0.000
0.623	0.00	0.000	0.000	0.000	0.000	0.623	0.000	0.000	0.000	0.00	0.00	0.474
0.000	0.46	0.000	0.349	0.000	0.000	0.000	0.000	0.349	0.000	0.46	0.46	0.349

Natural Language Processing Bootcamp

- TF-IDF allows us to understand the context of words across an entire corpus of documents, instead of just its relative importance in a single document.
- Coming up next we'll explore how to perform these operations with Python and SciKit-Learn!

Feature Extraction From Text

Continued

Natural Language Processing Bootcamp

- In this lecture we will learn:
  - A basic manual implementation of building a vocabulary
  - Using Scikit-learn for vectorization
  - Using Pipelines with Scikit-Learn

Feature Extraction From Text

Part Three - Code Along

Text Classification Code Along Project

Part One

Natural Language Processing Bootcamp

- Now that we understand the general machine learning process, classification metrics, and scikit-learn, let’s combine all these concepts by coding along with a real text data set!