Machine Learning and NLP

Machine Learning and NLP

- Machine Learning Review
  - Supervised vs. Unsupervised Learning, Classification vs. Regression
  - Simple Classification Example

- Text Classification Examples
  - Logistic Regression
  - Naive Bayes
  - Comparing Methods: Classification Metrics

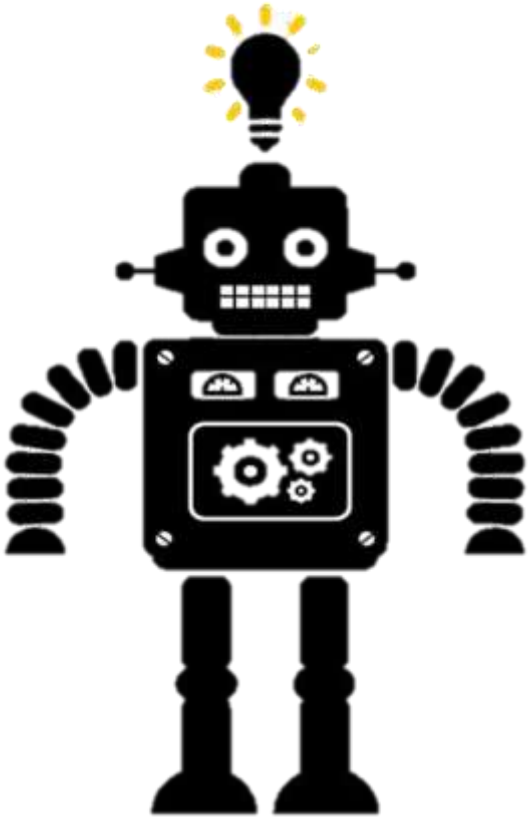
Machine Learning and NLP

- Machine Learning Review
  - Supervised vs. Unsupervised Learning
  - Classification vs. Regression

- Text Classification Examples
  - Logistic Regression
  - Naive Bayes
  - Comparing Methods: Classification Metrics

What is Machine Learning?

Machine learning allows computers to learn and infer from data by applying algorithms to observed data and make predictions based on the data.



Machine Learning Vocabulary

- **Target/Outcome:** predicted category or value of the data (column to predict)
- **Features:** properties of the data used for prediction (non-target columns)
- **Example:** a single data point within the data (one row)
- **Label:** the target value for a single data point

Types of Machine Learning

Supervised

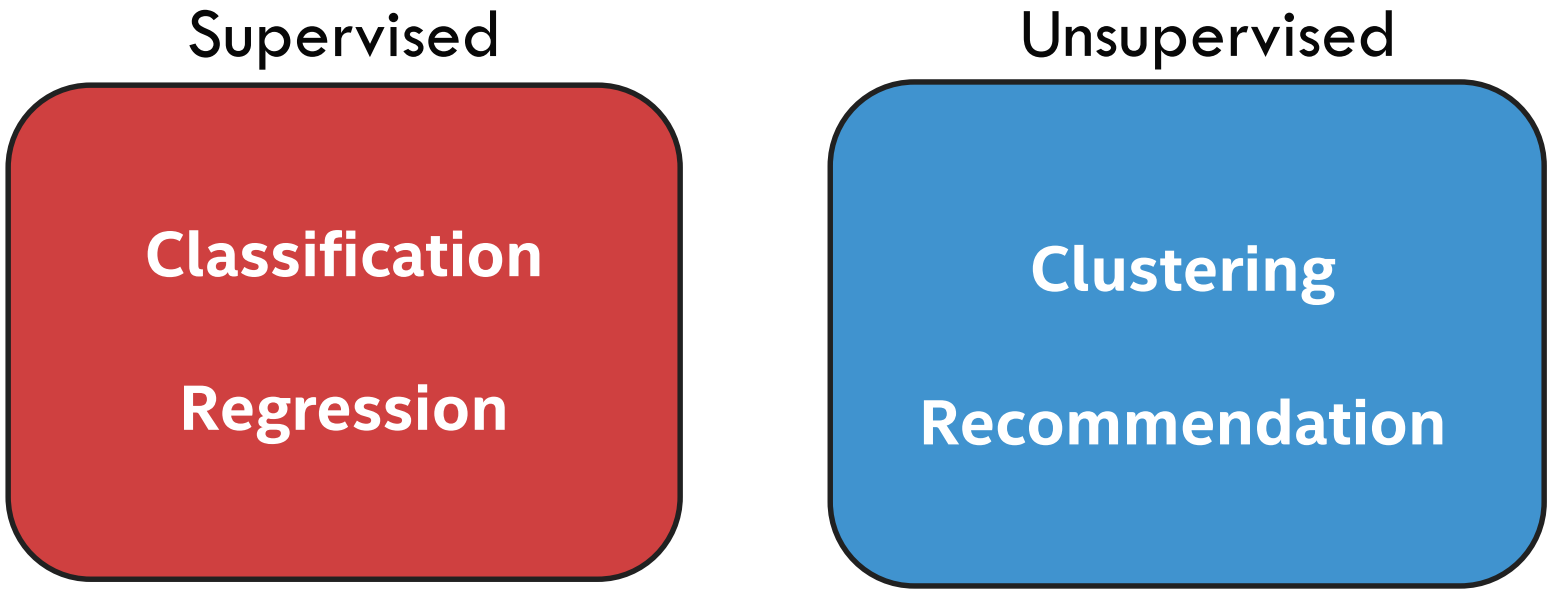
Here, the data points have known outcome. We train the model with data. We feed the model with correct answers. Model Learns and finally predicts new data's outcome.

Unsupervised

Here, the data points have unknown outcome. Data is given to the model. Right answers are not provided to the model. The model makes sense of the data given to it.

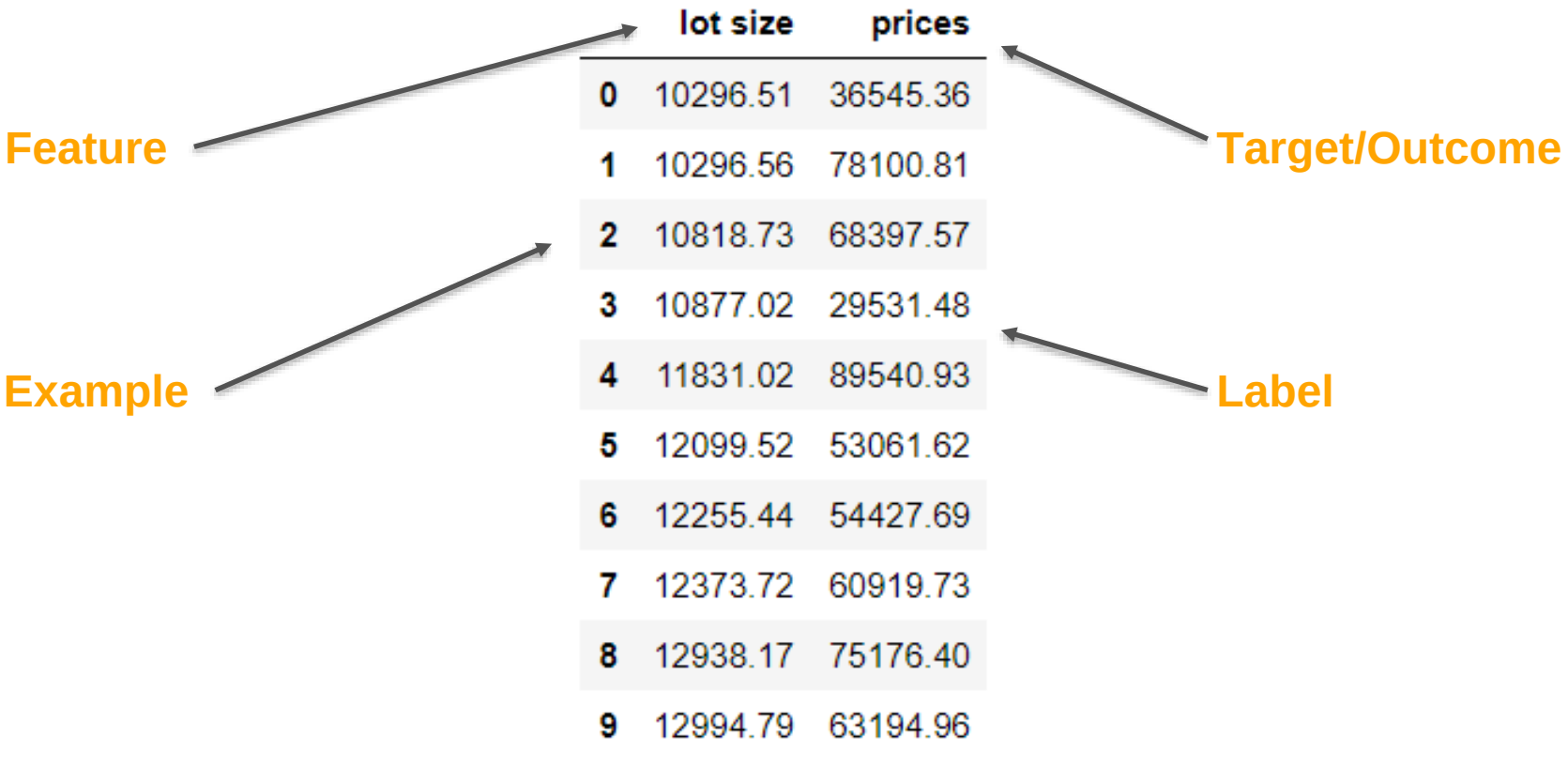
Can teach you something you were probably not aware of in the given dataset.

Types of Supervised And Unsupervised Learning



Supervised Learning

Supervised Learning Example: Housing Prices



	lot size	prices
0	10296.51	36545.36
1	10296.56	78100.81
2	10818.73	68397.57
3	10877.02	29531.48
4	11831.02	89540.93
5	12099.52	53061.62
6	12255.44	54427.69
7	12373.72	60919.73
8	12938.17	75176.40
9	12994.79	63194.96

Supervised Learning Example: Housing Prices

Supervised Learning Example: Housing Prices

Machine Learning and NLP

- Machine Learning Review
  - Supervised vs. Unsupervised Learning
  - Classification vs. Regression

- Text Classification Examples
  - Logistic Regression
  - Naive Bayes
  - Comparing Methods: Classification Metrics

Types of Supervised Learning

Regression outcome is continuous (numerical)

Types of Supervised Learning

Regression — outcome is continuous (numerical)

Classification — outcome is a category

Regression vs Classification Problems

Some supervised learning questions we can ask regarding movie data:

- Regression Questions
  - Can you predict the gross earnings for a movie? *$1.1 billion*
- Classification Questions
  - Can you predict whether a movie will win an Oscar or not? *Yes | No*
  - Can you predict what the MPAA rating is for a movie? *G | PG | PG-13 | R*

Regression

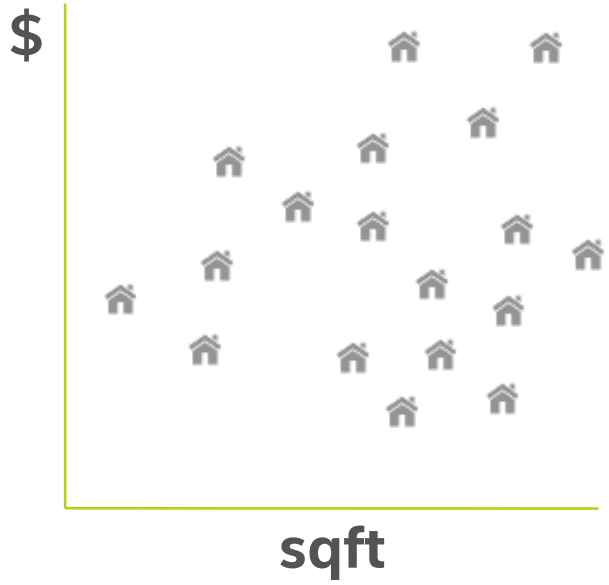
Predict a real numeric value for an entity with a given set of features.

Property Attributes

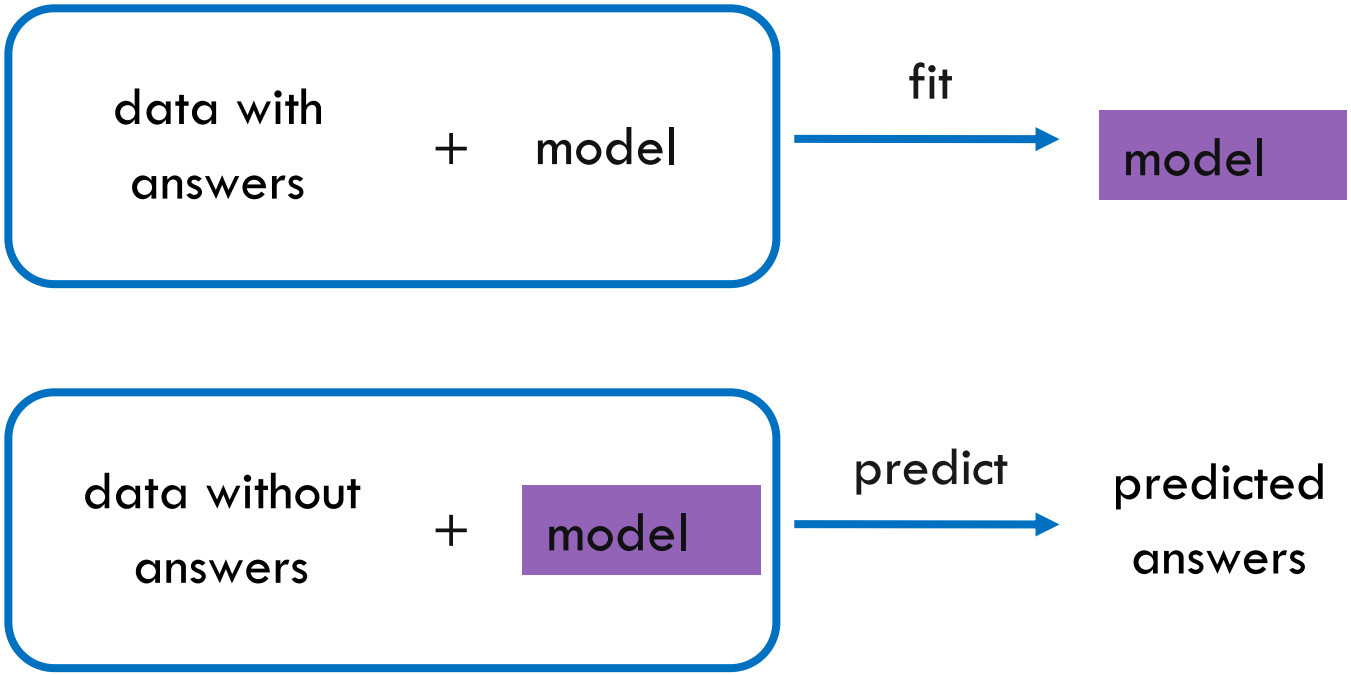
Price
Address
Type
Age
Parking
School
Transit

Total sqft
Lot Size
Bathrooms
Bedrooms
Yard
Pool
Fireplace

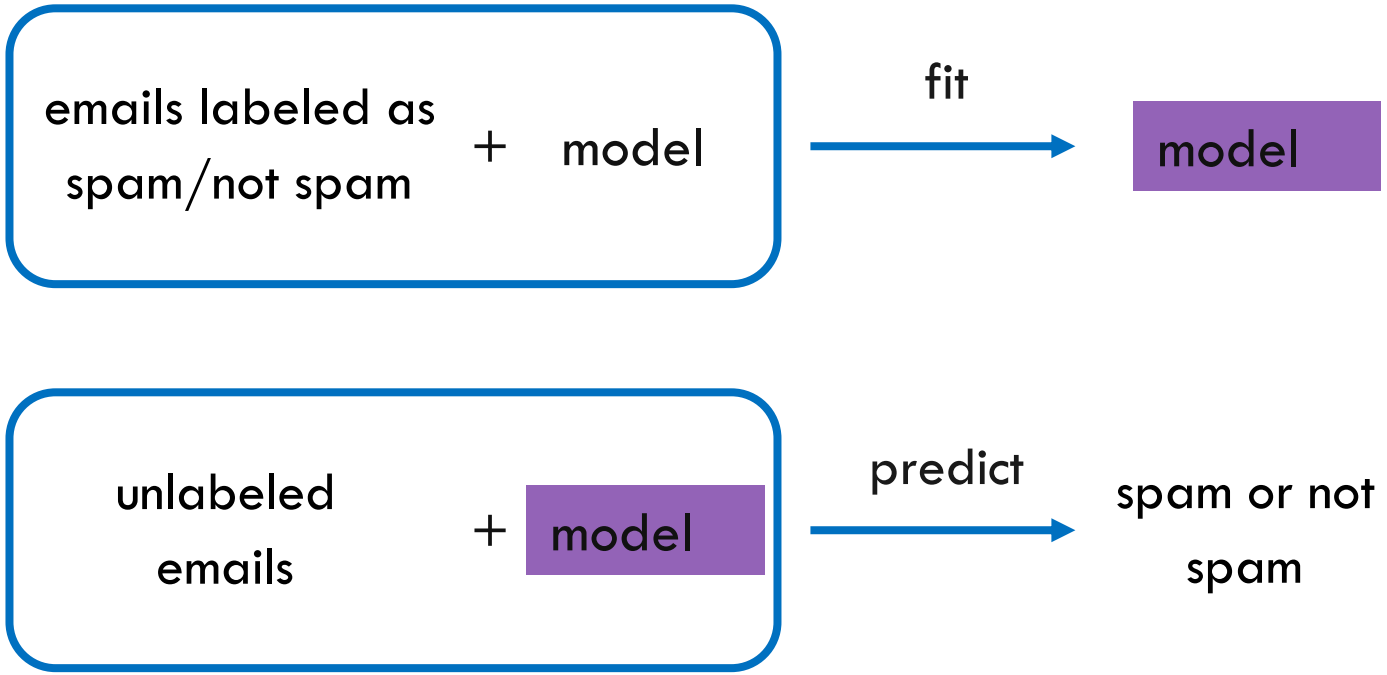
Linear Regression Model



Supervised Learning Overview



Classification: Categorical Answers



Machine Learning and NLP

- Machine Learning Review
  - Supervised vs. Unsupervised Learning, Classification vs. Regression
  - Classification Example

- Text Classification Examples
  - Logistic Regression
  - Naive Bayes
  - Comparing Methods: Classification Metrics

Machine Learning and NLP

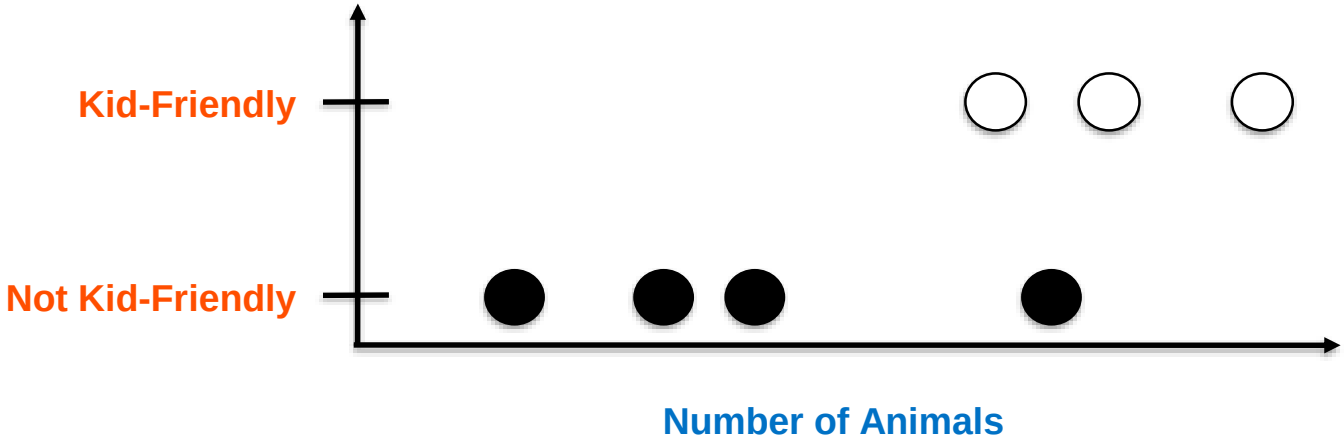
- Machine Learning Review
  - Supervised vs. Unsupervised Learning, Classification vs. Regression
  - Classification Example

- Text Classification Examples
  - Logistic Regression
  - Naive Bayes
  - Comparing Methods: Classification Metrics

Logistic Regression

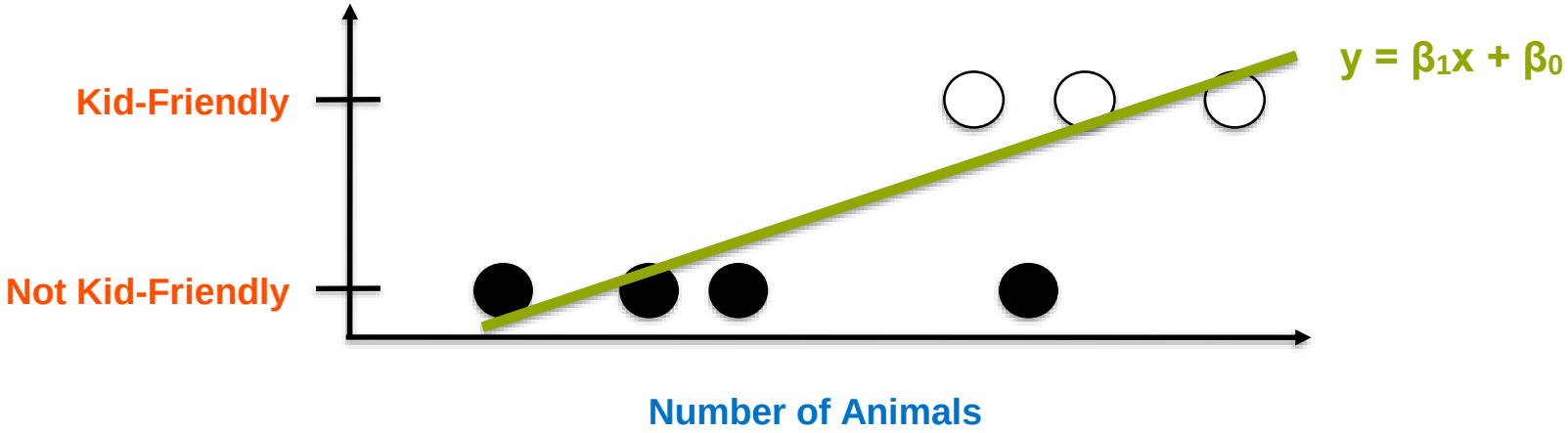
One of the most popular machine learning techniques for binary classification

Binary classification = how do you best split this data into two groups?



Logistic Regression

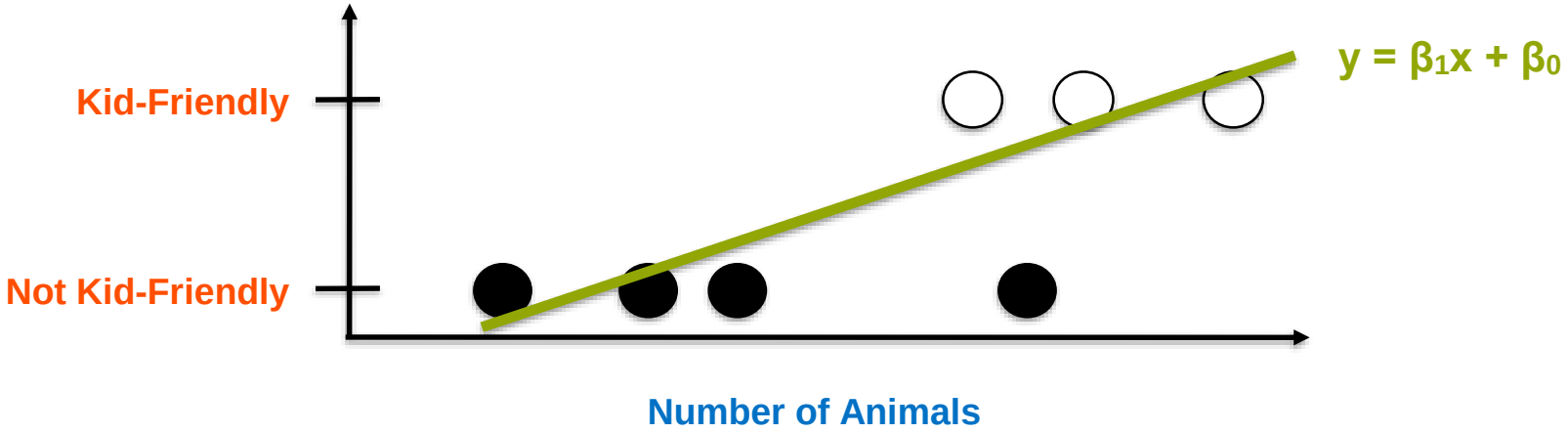
The most basic regression technique is linear regression



Logistic Regression

Problem: The y values of the line go from -∞ to +∞

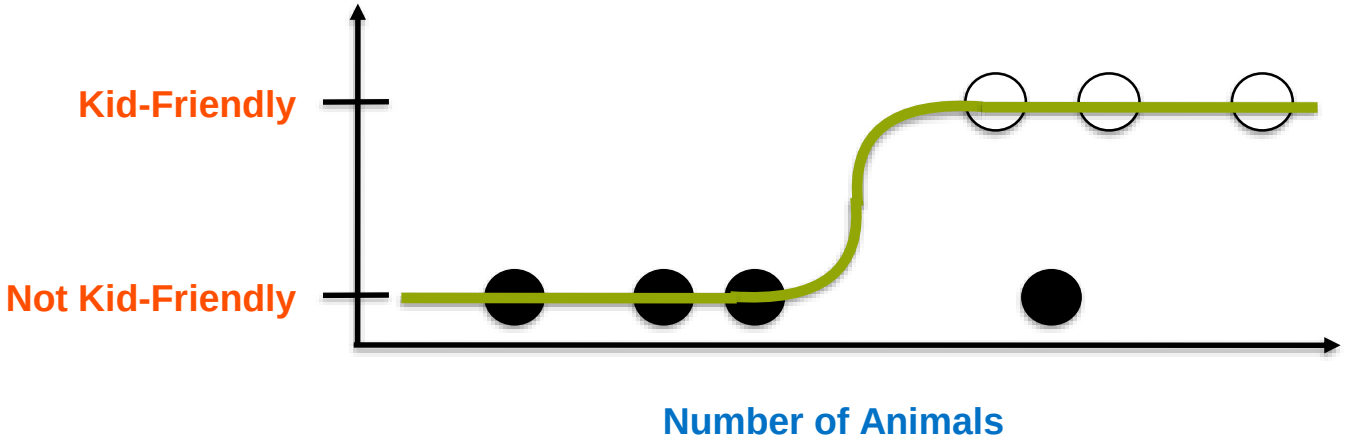
Let's try applying a transformation to the line to limit the y values from 0 to 1



Logistic Regression

The sigmoid function (aka the “S” curve) solves this problem

If you input the Number of Animals (x), the equation gives you the probability that the movie is Kid-Friendly (p), which is a number between 0 and 1

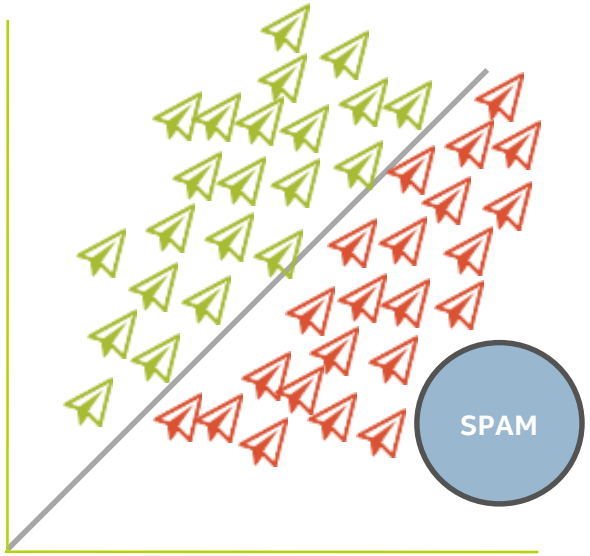


$$p = \frac{1}{1 + e^{-(\beta 1 x + \beta 0)}}$$

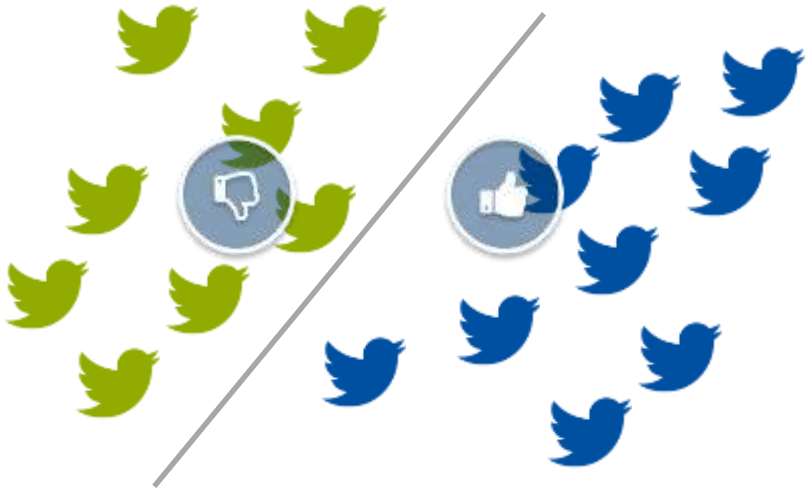
Logistic Regression examples

Predict a label for an entity with a given set of features.

Prediction



Sentiment Analysis



Building a Logistic Regression model

Steps for classification with NLP

1. *Prepare the data*: Read in labelled data and preprocess the data
2. *Split the data*: Separate inputs and outputs into a training set and a test set, repsectively
3. *Numerically encode inputs*: Using Count Vectorizer or TF-IDF Vectorizer
4. *Fit a model*: Fit a model on the training data and apply the fitted model to the test set
5. *Evaluate the model*: Decide how good the model is by calculating various error metrics

Step 1: Prepare the data

A classic use of text analytics is to flag messages as spam

Below is data from the SMS Spam Collection Data, which is a set of over 5K English text messages that have been labeled as spam or ham (legitimate)

Text Message	Label
Nah I don't think he goes to usf, he lives around here though	ham
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	spam
WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.	spam
I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.	ham
I HAVE A DATE ON SUNDAY WITH WILL!!	ham
…	…

Step 1: Prepare the data [Code]

Input:

```
# make sure the data is labeled
import pandas as pd

data = pd.read_table('SMSSpamCollection.txt', header=None)
data.columns = ['label', 'text']
print(data.head()) # print function requires Python 3
```

Output:

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

Step 1: Prepare the data [Code]

Input:

```
# remove words with numbers, punctuation and capital letters
import re
import string

alphanumeric = lambda x: re.sub(r"""\w*\d\w*""", ' ', x)
punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x.lower())

data['text'] = data.text.map(alphanumeric).map(punc_lower)
print(data.head())
```

Output:

	label	text
0	ham	go until jurong point crazy available only ...
1	ham	ok lar joking wif u oni
2	spam	free entry in a wkly comp to win fa cup fina...
3	ham	u dun say so early hor u c already then say
4	ham	nah i don t think he goes to usf he lives aro...

Step 2: Split the data (into inputs and outputs)

To fit a model, the data needs to be split into inputs and outputs

The inputs and output of these models have various names

- Inputs: Features, Predictors, Independent Variables, X's
- Outputs: Outcome, Response, Dependent Variable, Y

#	label	congrats	eat	tonight	winner	chicken	dinner	wings
0	ham	0	1	0	0	0	0	0
1	ham	0	1	1	0	0	0	0
2	spam	0	0	0	1	0	0	0
.	…	…	…	…	…	…	…	…

Step 2: Split the data [Code]

Input:

```
# split the data into inputs and outputs
X = data.text # inputs into model
y = data.label # output of model
```

Output:

```
X.head()
```

```
0    go until jurong point  crazy   available only ...
1                        ok lar    joking wif u oni
2    free entry in   a wkly comp to win fa cup fina...
3    u dun say so early hor    u c already then say
4    nah i don t think he goes to usf  he lives aro...
Name: text, dtype: object
```

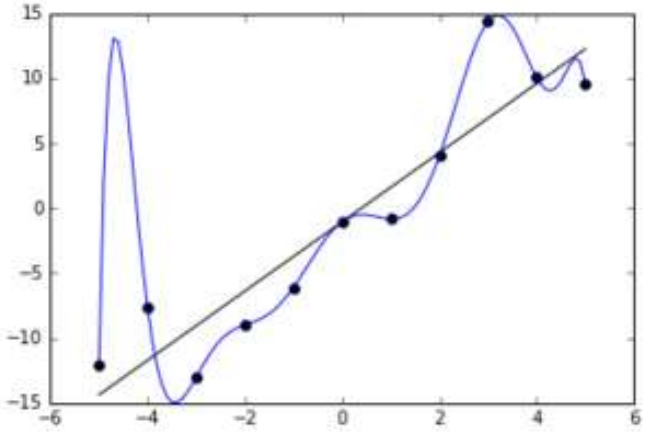
```
y.head()
```

```
0     ham
1     ham
2    spam
3     ham
4     ham
Name: label, dtype: object
```

Step 2: Split the data (into a training and test set)

Why do we need to split data into training and test sets?

- Let's say we had a data set with 100 observations and we found a model that fit the data *perfectly*
- What if you were to use that model on a brand new data set?



Blue = Overfitting

Black = Correct

Source: https://en.wikipedia.org/wiki/Overfitting

Step 2: Split the data (into a training and test set)

To prevent the issue of overfitting, we divide observations into two sets

- A model is fit on the training data and it is evaluated on the test data
- This way, you can see if the model generalizes well

	label	congrats	eat	tonight	winner	chicken	dinner	wings
0	ham	0	1	0	0	0	0	0
1	ham	0	1	1	0	0	0	0
2	spam	0	0	0	1	0	0	0
3	spam	1	0	0	0	0	0	0
4	ham	0	0	0	0	0	1	0
5	ham	0	0	1	0	0	0	0
6	ham	0	0	0	0	0	0	0
7	spam	0	0	0	0	0	0	0
8	ham	0	0	0	0	0	1	0
9	ham	0	0	0	0	1	1	0
10	spam	0	0	0	0	0	0	0
11	ham	0	0	0	0	0	0	1

Training Set (70-80%)

Test Set (20-30%)

Step 2: Split the data [Code]

Input:

```
# split the data into a training and test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# test size = 30% of observations, which means training size = 70% of observations
# random state = 42, so we all get the same random train / test split
```

Output:

```
X_train.head()
```

```
708      quite late lar    ard   anyway i wun b drivin
4338                        on a tuesday night r u   real
5029    go chase after her and run her over while she ...
4921     g says you never answer your texts  confirm deny
2592         still work going on  it is very small house
Name: text, dtype: object
```

```
X_train.shape
```

```
(3900,)
```

```
y_train.head()
```

```
708     ham
4338    ham
5029    ham
4921    ham
2592    ham
Name: label, dtype: object
```

```
y_train.shape
```

```
(3900,)
```

```
X_test.shape
```

```
(1672,)
```

```
y_test.shape
```

```
(1672,)
```

Step 3: Numerically encode the input data [Code]

Input:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words='english')

X_train_cv = cv.fit_transform(X_train) # fit_transform learns the vocab and one-hot
encodes
X_test_cv  = cv.transform(X_test) # transform uses the same vocab and one-hot encodes

# print the dimensions of the training set (text messages, terms)
print(X_train_cv.toarray().shape)
```

Output:

```
(3900, 6103)
```

Step 4: Fit model and predict outcomes [Code]

Input:

```
# Use a logistic regression model
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()

# Train the model
lr.fit(X_train_cv, y_train)

# Take the model that was trained on the X_train_cv data and apply it to the X_test_cv
data
y_pred_cv = lr.predict(X_test_cv)
y_pred_cv # The output is all of the predictions
```

Output:

```
array(['ham', 'ham', 'ham', ..., 'ham', 'spam', 'ham'], dtype=object)
```

Step 5: Evaluate the model

After fitting a model on the training data and predicting outcomes for the test data, how do you know if the model is a good fit?

#	Actual	Predicted	Result
1	ham	ham	true negative
2	ham	ham	true negative
3	spam	spam	true positive
4	spam	spam	true positive
5	ham	ham	true negative
6	ham	spam	false positive
7	ham	ham	true negative
8	ham	ham	true negative
9	ham	ham	true negative
10	spam	spam	true positive

Confusion Matrix

Actual \ Predicted	ham	spam
ham	True Negative (6)	False Positive (1)
spam	False Negative (0)	True Positive (3)

Step 5: Evaluate the model

After fitting a model on the training data and predicting outcomes for the test data, how do you know if the model is a good fit?

Error Metrics

- Accuracy = (TP + TN) / All
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 Score = 2*(P*R)/(P+R)

P = Precision R = Recall

Confusion Matrix

Actual \ Predicted	ham	spam
ham	True Negative (6)	False Positive (1)
spam	False Negative (0)	True Positive (3)

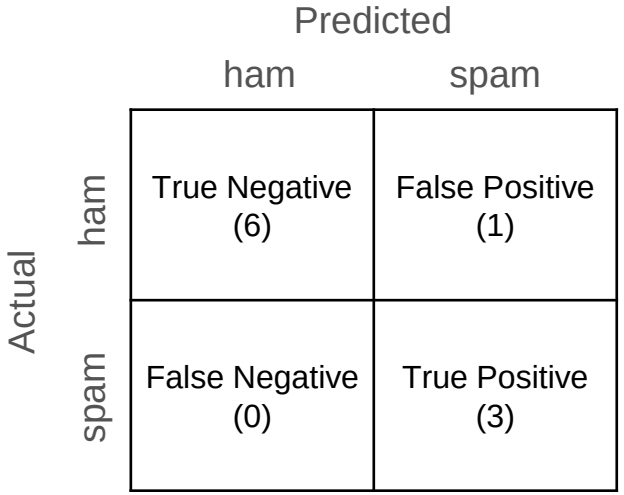
Step 5: Evaluate the model

After fitting a model on the training data and predicting outcomes for the test data, how do you know if the model is a good fit?

Error Metrics

- Accuracy = (TP + TN) / All = 0.9
- Precision = TP / (TP + FP) = 0.75
- Recall = TP / (TP + FN) = 1
- F1 Score = 2*(P*R)/(P+R) = 0.86

Confusion Matrix



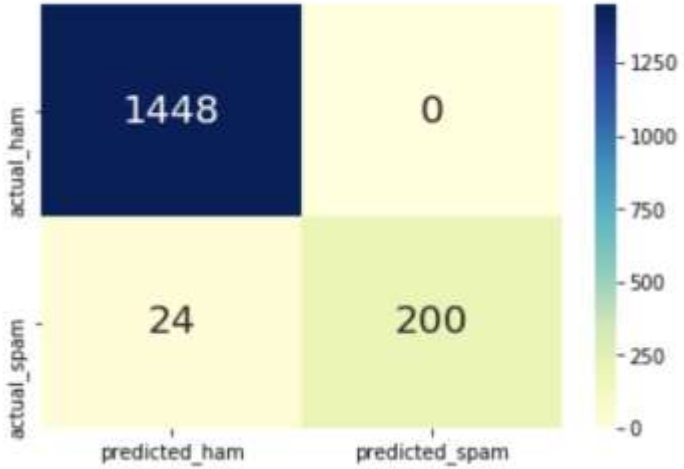
Step 5: Evaluate the model [Code]

Input:

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

cm = confusion_matrix(y_test, y_pred_cv)
sns.heatmap(cm, xticklabels=['predicted_ham', 'predicted_spam'], yticklabels=['actual_ham', 'actual_spam'],
annot=True, fmt='d', annot_kws={'fontsize':20}, cmap="YlGnBu");

true_neg, false_pos = cm[0]
false_neg, true_pos = cm[1]

accuracy = round((true_pos + true_neg) / (true_pos + true_neg + false_pos + false_neg),3)
precision = round((true_pos) / (true_pos + false_pos),3)
recall = round((true_pos) / (true_pos + false_neg),3)
f1 = round(2 * (precision * recall) / (precision + recall),3)

print('Accuracy: {}'.format(accuracy))
print('Precision: {}'.format(precision))
print('Recall: {}'.format(recall))
print('F1 Score: {}'.format(f1))
```

Step 5: Evaluate the model [Code]

Output:




```
Accuracy: 0.986
Precision: 1.0
Recall: 0.893
F1 Score: 0.943
```

Logistic Regression Checkpoint

What was our original goal?

To classify text messages as spam or ham

How did we do that?

By collecting labeled data, cleaning the data, splitting it into a training and test set, numerically encoding it using Count Vectorizer, fitting a logistic regression model on the training data, and evaluating the results of the model applied to the test set

Was it a good model?

It seems good, but let's see if we can get better metrics with another classification technique, Naive Bayes

Machine Learning and NLP

- Machine Learning Review
  - Supervised vs. Unsupervised Learning,
  - Classification vs. Regression

- Text Classification Examples
  - Logistic Regression
  - Naive Bayes
  - Comparing Methods: Classification Metrics

Naive Bayes

One of the simpler and less computationally intensive techniques

Naive Bayes tends to perform well on text classifications

1. Conditional Probability and Bayes Theorem
2. Independence and Naive Bayes
3. Apply Naive Bayes to Spam Example and Compare with Logistic Regression

Naive Bayes: Bayes Theorem

Conditional Probability = what's the probability that something will happen, given that something else has happened?

Spam Example = what's the probability that this text message is spam, given that it contains the word "cash"?

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(\text{spam} \mid \text{cash}) = \frac{P(\text{cash} \mid \text{spam}) \times P(\text{spam})}{P(\text{cash})}$$

Naive Bayes: Independent Events

Naive Bayes assumes that each event is independent, or that it has no effect on other events. This is a naive assumption, but it provides a simplified approach.

Spam Example: Naive Bayes assumes that each word in a spam message (like “cash” and “winner”) is unrelated. This is unrealistic, but it’s the naive assumption.

The higher probability wins

P(spam | winner of some cash) =

P(winner | spam) x P(of | spam) x P(some | spam) x P(cash | spam) x P(spam)

P(ham | winner of some cash) =

P(winner | ham) x P(of | ham) x P(some | ham) x P(cash | ham) x P(ham)

Naive Bayes: Fit model [Code]

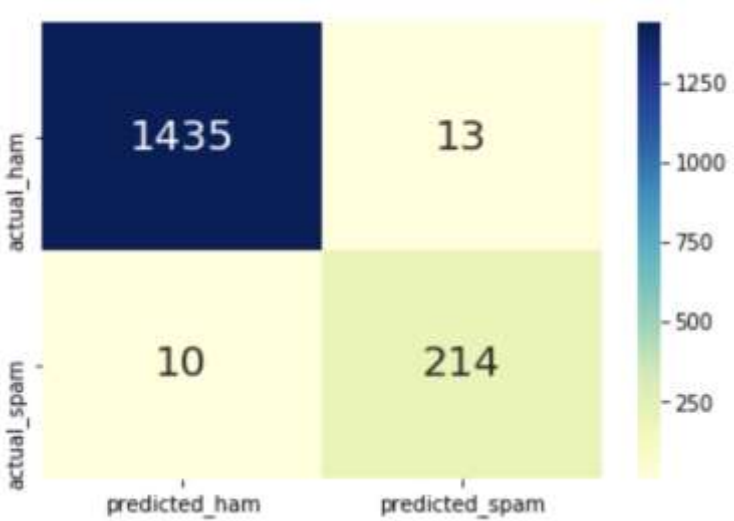
Input:

```
# Use a Naive Bayes model
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()

# Train the model
nb.fit(X_train_cv, y_train)

# Take the model that was trained on the X_train_cv data and apply it to the X_test_cv
data
y_pred_cv_nb = nb.predict(X_test_cv)
y_pred_cv_nb # The output is all of the predictions
```

Output:

```
array(['ham', 'ham', 'ham', ..., 'ham', 'spam', 'ham'], dtype='<U4')
```

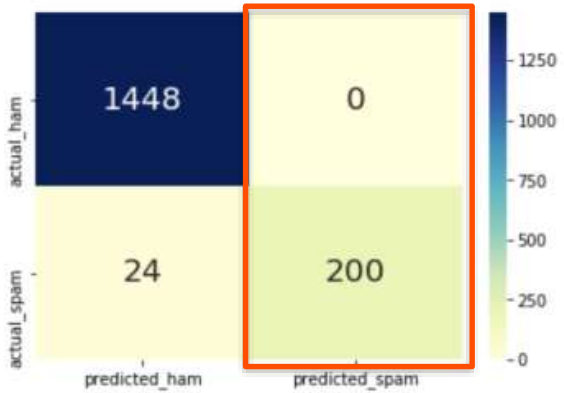
Naive Bayes: Results



Accuracy: 0.986
Precision: 0.939
Recall: 0.952
F1 Score: 0.945

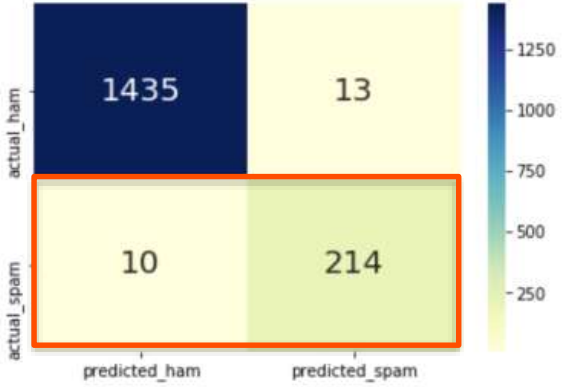
Compare Logistic Regression and Naive Bayes

Logistic Regression



Accuracy: 0.986
Precision: 1.0
Recall: 0.893
F1 Score: 0.943

Naive Bayes



Accuracy: 0.986
Precision: 0.939
Recall: 0.952
F1 Score: 0.945

Machine Learning and NLP Review

- Machine Learning Review
  - Supervised Learning > Classification Techniques > Logistic Regression & Naive Bayes
  - Error Metrics for Classification > Accuracy | Precision | Recall | F1 Score

- Classification with NLP
  - Make sure the data is labeled and split into a training and test set
  - Clean the data and use one-hot encoding to put in a numeric format for modeling
  - Fit the model on the training data, apply the model to the test data and evaluate

Software