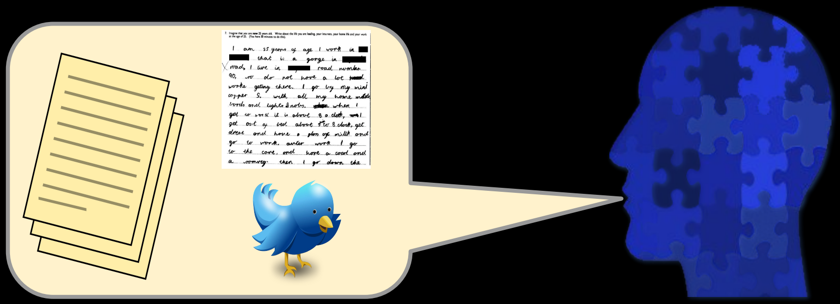
Text Classification:
Lexicon-Based and Supervised Logistic Regression

CSE354 - Spring 2021

NLP's practical applications

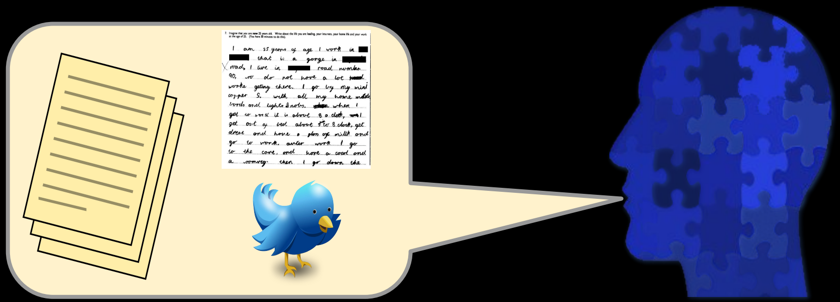


- Machine translation
- Automatic speech recognition
  - Personalized assistants
  - Auto customer service
- Information Retrieval
  - Web Search
  - Question Answering
- Text Categorization:
  e.g. Sentiment Analysis
- Computational Social Science

how? →

- Machine learning:
  - Logistic regression
  - Probabilistic modeling
  - Recurrent Neural Networks
  - Transformers
- Algorithms, e.g.:
  - Graph analytics
  - Dynamic programming
- Data science
  - Hypothesis testing

NLP's practical applications



- Machine translation
- Automatic speech recognition
- Personalized assistants
- Auto customer service
- Information Retrieval
- Web Search
- Question Answering
- **Text Categorization: e.g. Sentiment Analysis**
- Computational Social Science

how?

- **Lexica**
- Machine learning:
  - **Logistic regression**
  - Probabilistic modeling
  - Recurrent Neural Networks
  - Transformers
- Algorithms, e.g.:
  - Graph analytics
  - Dynamic programming
- Data science
  - Hypothesis testing

Topics we will cover

- *Lexicon-based classification (closed-vocabulary)*
- *Supervised classification (open-vocabulary)*
  - Goal of logistic regression
  - The “loss function” -- what logistic regression tries to optimize
  - Logistic regression with multiple features
  - How to evaluation: Training and test datasets
  - Overfitting: role of regularization

Text Classification

The Buccaneers win it!

President Biden vetoed bill

Twitter to be acquired by Apple

I like the the movie.

The movie is like terrible.

Sentiment

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)

Sentiment

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..."

“This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone. We called each other when we got home. The voice on my phone was not so clear, worse than my previous phone. ..." (Liu, 2010)

Lexica

Lin, C., & He, Y. (2009, November). Joint sentiment/topic model for sentiment analysis. In Proceedings of the 18th ACM conference on Information and knowledge management (pp. 375-384).

Table 1: Paradigm word list.

Positive	dazzling brilliant phenomenal excellent fantastic gripping mesmerizing riveting spectacular cool awesome thrilling moving exciting love wonderful best great superb still beautiful
Negative	sucks terrible awful unwatchable hideous bad cliched boring stupid slow worst waste unexcit rubbish tedious unbearable pointless cheesy frustrated awkward disappointing

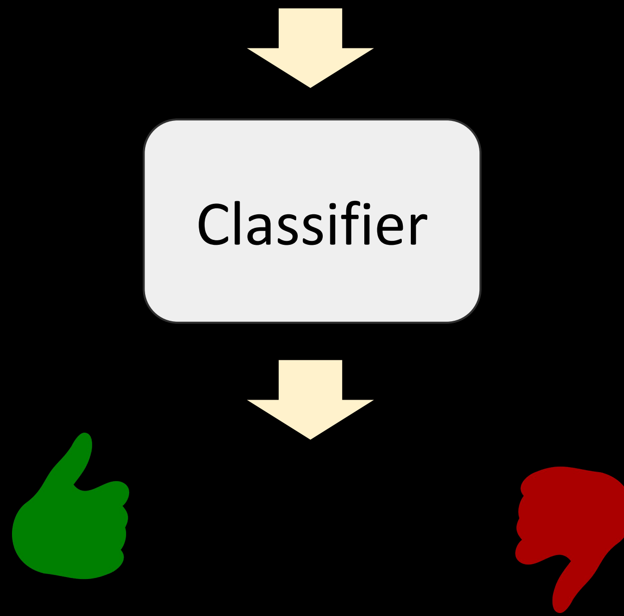
	Proposed word lists	Accuracy	Ties
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58%	75%
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64%	39%

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

Pang, B. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002.

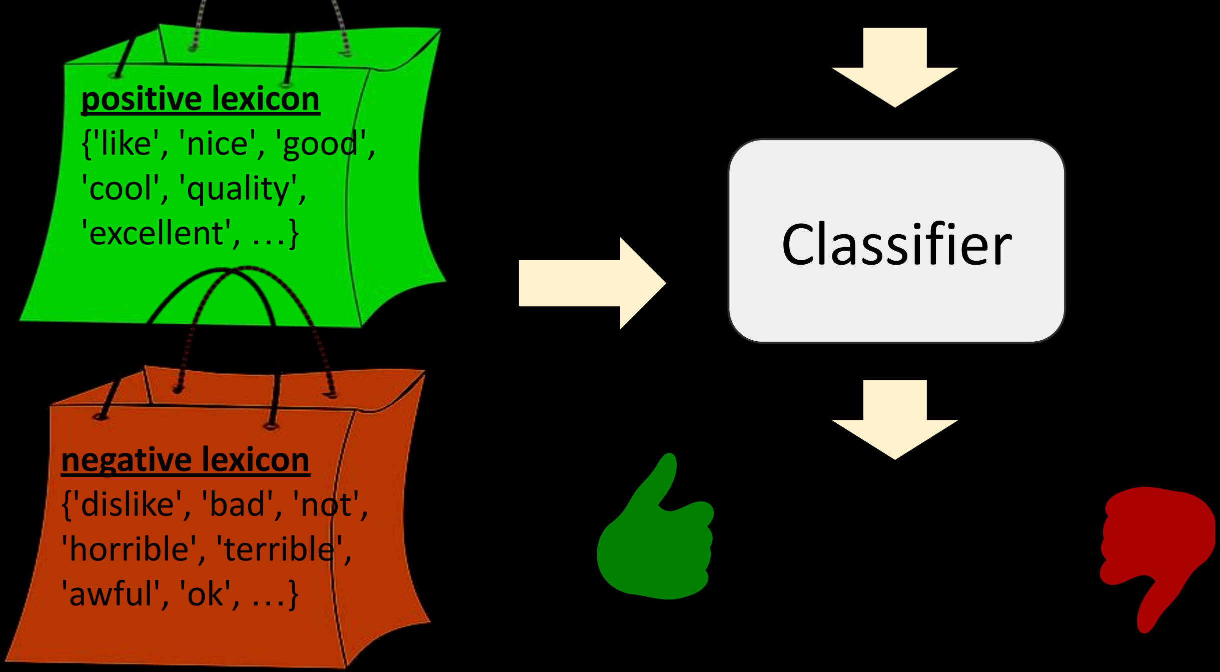
Sentiment Analysis

"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)



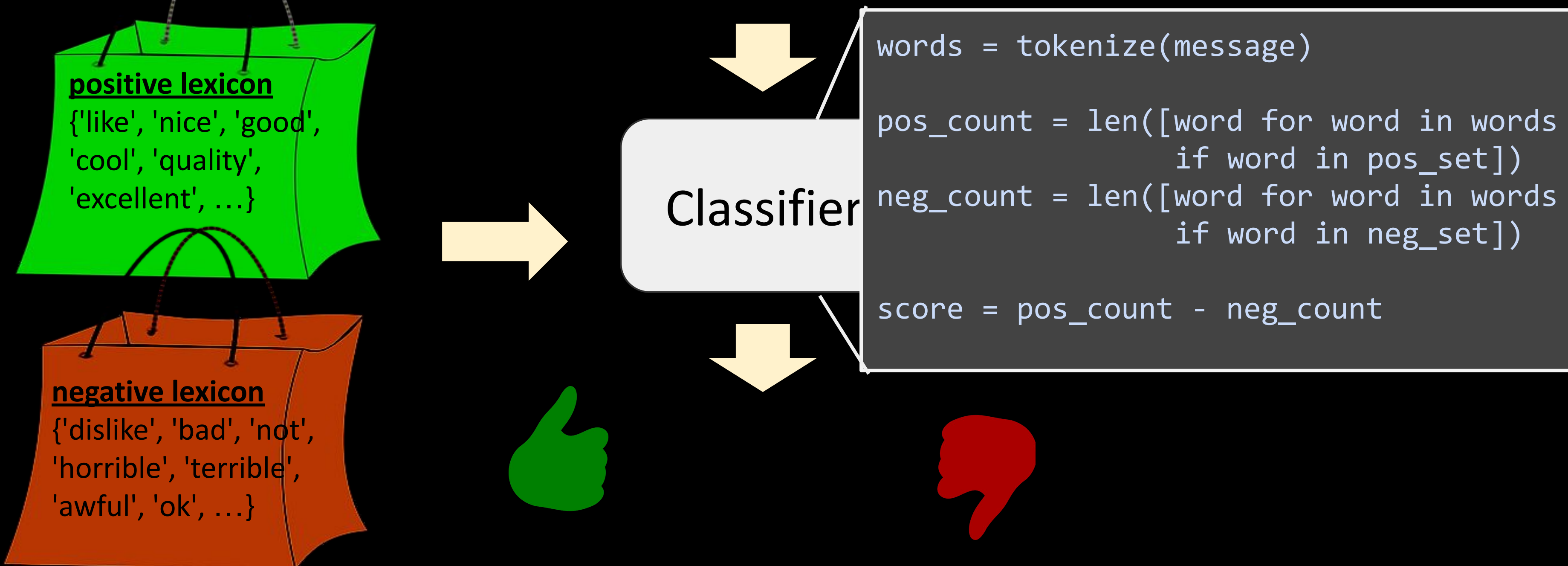
Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)



Sentiment Analysis

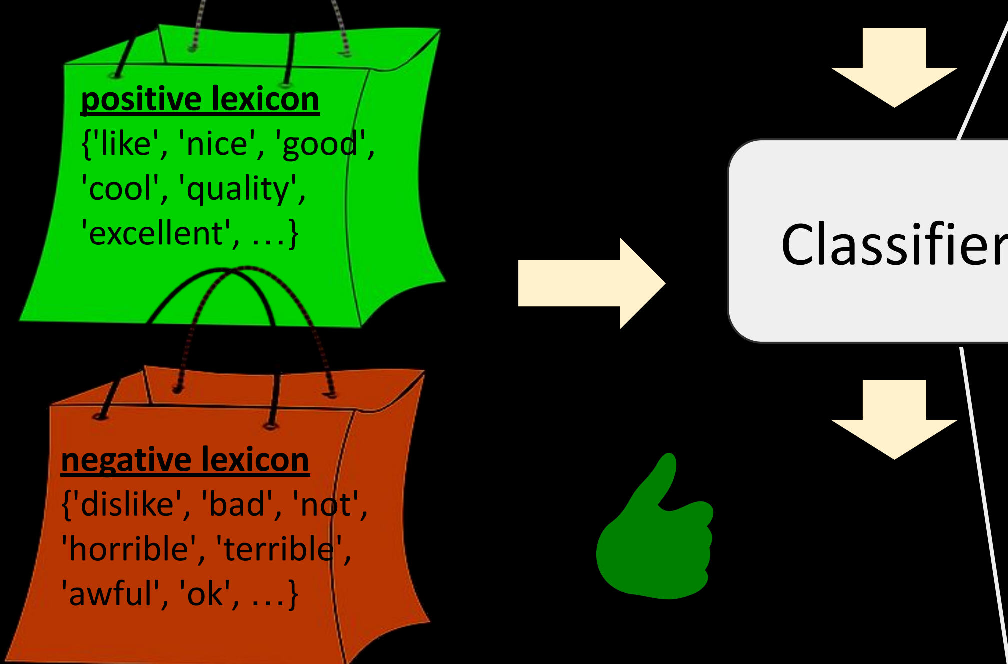
"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)




```
words = tokenize(message)

pos_count = len([word for word in words
                 if word in pos_set])
neg_count = len([word for word in words
                 if word in neg_set])

score = pos_count - neg_count
```

Sentiment Analysis

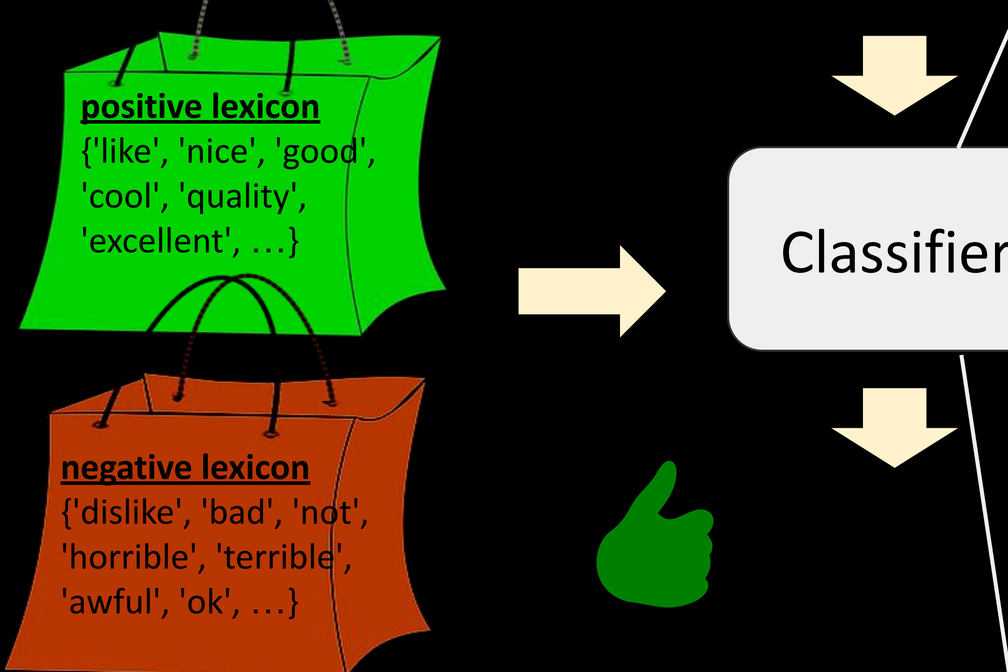
"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)




```
words = tokenize(message)

pos_count = len([word for word in words
                 if word in pos_set])
neg_count = len([word for word in words
                 if word in neg_set])

pos_p = pos_count/len(words)
neg_p = neg_count/len(words)

score = pos_p - neg_p

if (score > 0): return "positive"
else: return "negative"
```

Sentiment Analysis

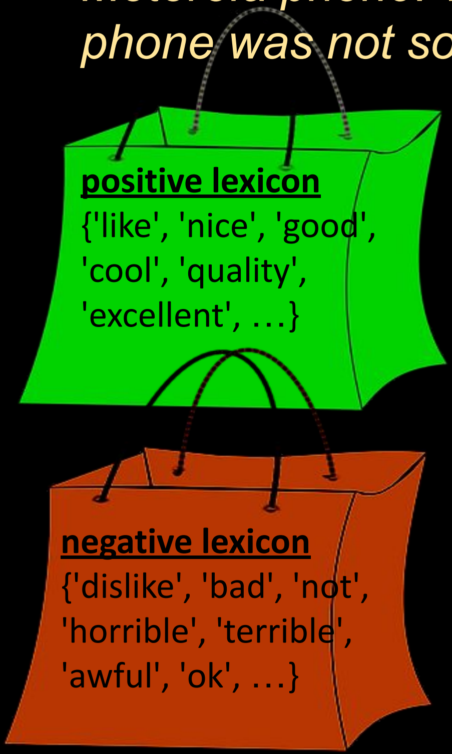
"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)

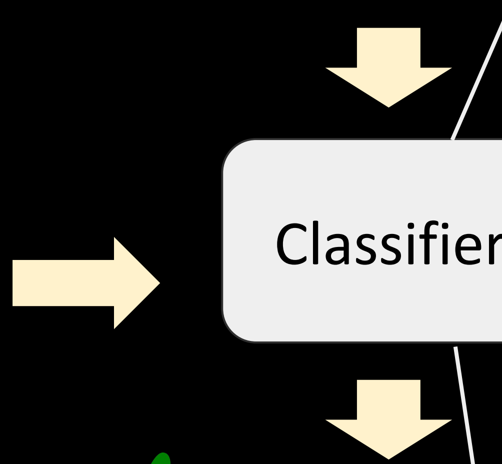



```
words = tokenize(message)

pos_count = len([word for word in words
                 if word in pos_set])
neg_count = len([word for word in words
                 if word in neg_set])

pos_p = pos_count/len(words)
neg_p = neg_count/len(words)

score = pos_p - neg_p

if (score > thresh): return "positive"
else: return "negative"
```

Sentiment Analysis

“This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone. We called each other when we got home. The voice on my phone was not so clear, worse than my previous phone. ..."





```
words = tokenize(message)

pos_count = len([word for word in words
                 if word in pos_set])
neg_count = len([word for word in words
                 if word in neg_set])

pos_p = pos_count/len(words)
neg_p = neg_count/len(words)

score = pos_p - neg_count

if (score > thresh): return "positive"
else: return "negative"
```

Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)

Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. However, my mother was mad with me as I did not tell her before I bought it. She also thought the phone was too expensive, and wanted me to return it to the shop." (Liu, 2010)

Sentiment Analysis

“This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone. We called each other when we got home. The voice on my phone was not so clear, worse than my previous phone. The camera was good. My girlfriend was quite happy with her phone. I wanted a phone with good voice quality. So my purchase was a real disappointment. I returned the phone yesterday."(Liu, 2010)

Sentiment -- Using Statistics

	Proposed word lists	Accuracy	Ties
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58%	75%
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64%	39%

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

	Proposed word lists	Accuracy	Ties
Human 3 + stats	positive: *love, wonderful, best, great, superb, still, beautiful* negative: *bad, worst, stupid, waste, boring, ?, !*	69%	16%

Figure 2: Results for baseline using introspection and simple statistics of the data (including *test* data).

Pang, B. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002.

Sentiment -- Using Statistics

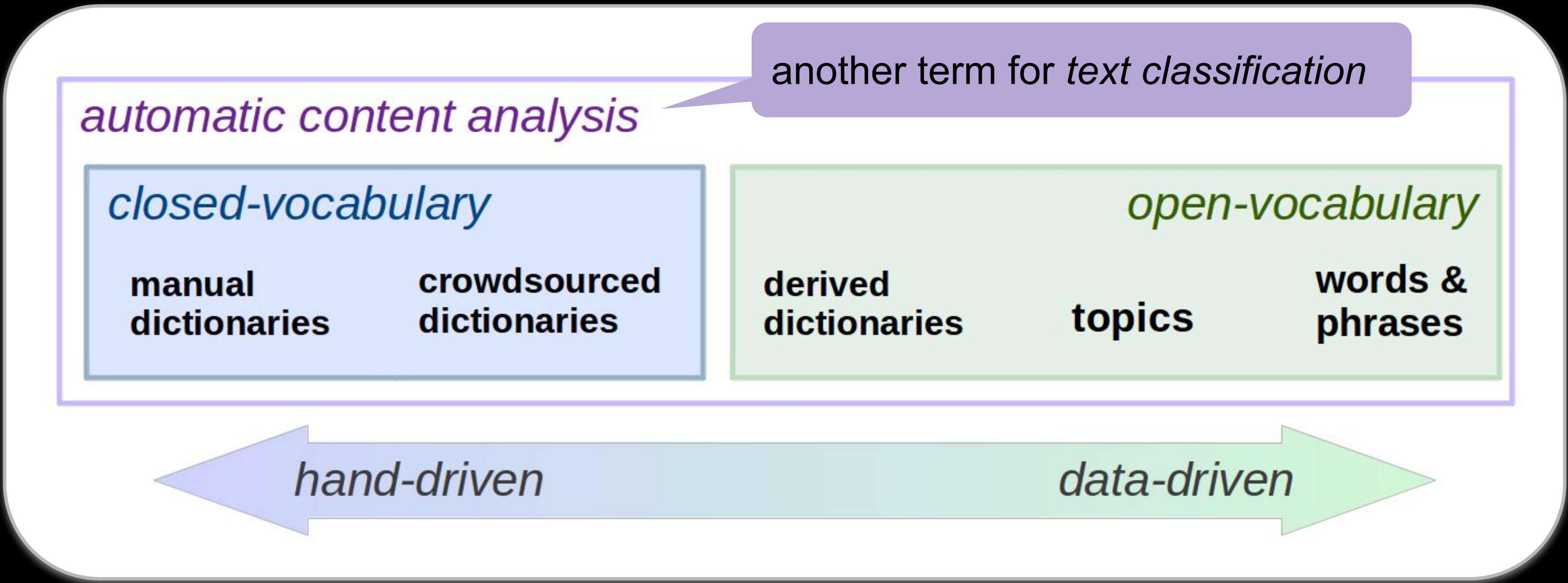
	Proposed word lists	Accuracy	Ties
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58%	75%
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64%	39%

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

	Proposed word lists	Accuracy	Ties
Human 3 + stats	positive: *love, wonderful, best, great, superb,* $\boxed{\textit{still,}}$ *beautiful* negative: *bad, worst, stupid, waste, boring,* $\boxed{?}$, $\boxed{!}$	69%	16%

Figure 2: Results for baseline using introspection and simple statistics of the data (including *test* data).

Pang, B. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002.



	Proposed word lists	Accuracy	Ties
Human 3 + stats	positive: *love, wonderful, best, great, superb, still, beautiful* negative: *bad, worst, stupid, waste, boring, ?, !*	69%	16%

Figure 2: Results for baseline using introspection and simple statistics of the data (including *test* data).

Supervised Classification

Supervised Classification

X - features of N observations (i.e. words)

Y - class of each of N observations

GOAL: Produce a *model* that outputs the most likely class y_i, given features x_i.

$$f(X) = Y$$

Supervised Classification

X - features of N observations (i.e. words)

Y - class of each of N observations

GOAL: Produce a *model* that outputs the most likely class y_i, given features x_i.

$f(X) = Y$

$$
\begin{array}{ccc}
i & X & Y \\
\begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} &
\begin{pmatrix} 0.0 \\ 0.5 \\ 1.0 \\ 0.25 \\ 0.75 \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}
\end{array}
$$

Supervised Classificati

X - features of N observations (

Y - class of each of N obse vatio

Some function or rules to go from *X* to *Y*, as close as possible.

GOAL: Produce a *model* that outputs the most likely class y_i, given features x_i.

$f(X) = Y$

$$\begin{array}{c c c} i & X & Y \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 0.0 \\ 0.5 \\ 1.0 \\ 0.25 \\ 0.75 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \end{array}$$

Supervised Classification

Supervised Machine Learning: Build a model with examples of outcomes (i.e. Y) that one is trying to predict. (The alternative, *unsupervised* machine learning, tries to learn with only an X).

Classification: The outcome (Y) is a discrete class.

for example: $y \in$ {noun, verb, adjective, adverb}

$y \in$ {positive_sentiment, negative_sentiment}).

Classification as Producing a Probability

Binary classification goal: Build a model that can estimate $P(A=1|B=?)$

i.e. given B, yield (or "predict") the probability that A=1

Classification as Producing a Probability

Binary classification goal: Build a “model” that can estimate $P(A=1|B=?)$

i.e. given B, yield (or “predict”) the probability that A=1

In machine learning, tradition to use **Y** for the variable being predicted and **X** for the features use to make the prediction.

Classification as Producing a Probability

Binary classification goal: Build a "model" that can estimate $P(Y=1|X=?)$

i.e. given X, yield (or "predict") the probability that $Y=1$

In machine learning, tradition is to use **Y** for the variable being predicted and **X** for the features use to make the prediction.

Classification as Producing a Probability

Binary classification goal: Build a "model" that can estimate P(Y=1|X=?)

i.e. given X, yield (or "predict") the probability that Y=1

In machine learning, tradition is to use **Y** for the variable being predicted and **X** for the features use to make the prediction.

Example: Y: 1 if target is verb, 0 otherwise;
X: 1 if "was" occurs before target; 0 otherwise

I was reading for NLP. *We were fine.* *I am good.*

The cat was very happy. *We enjoyed the reading material.* *I was good.*

Classification as Producing a Probability

Binary classification goal: Build a "model" that can estimate P(Y=1|X=?)

i.e. given X, yield (or "predict") the probability that Y=1

In machine learning, tradition is to use **Y** for the variable being predicted and **X** for the features use to make the prediction.

Example: Y: 1 if target is verb, 0 otherwise;
X: 1 if "was" occurs before target; 0 otherwise

I was reading for NLP. *We were fine.* *I am good.*

The cat was very happy. *We enjoyed the reading material.* *I was good.*

Classification as Producing a Probability

Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

Classification as Producing a Probability

Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

Classification as Producing a Probability

Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

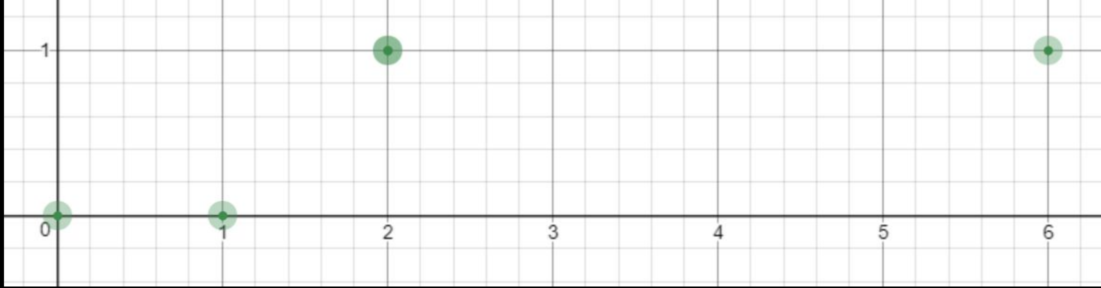
They attend Stony Brook University.   *Next to the brook Gandalf lay thinking.*

The trail was very stony.   *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

x	y
2	1
1	0
0	0
6	1
2	1

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

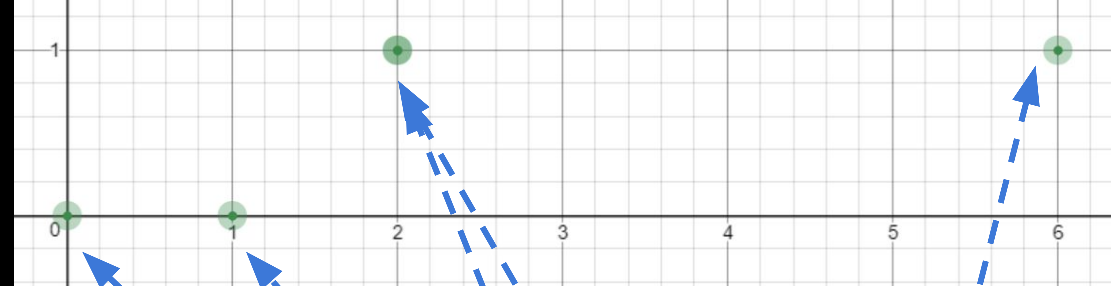
They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

x	y
2	1
1	0
0	0
6	1
2	1

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

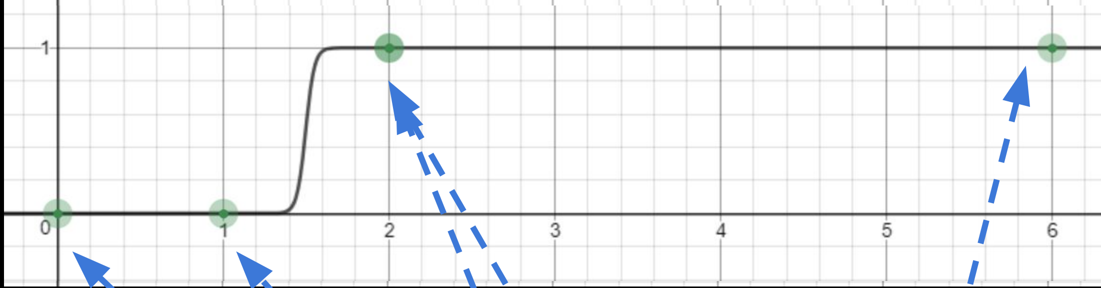
They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

x	y
2	1
1	0
0	0
6	1
2	1

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

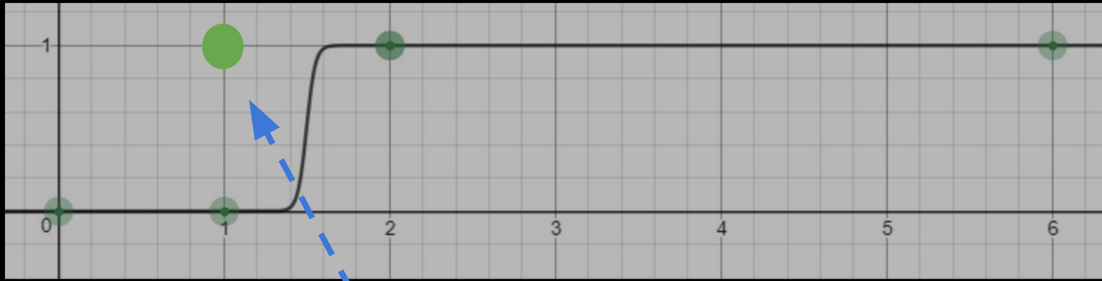
They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

x	y
2	1
1	0
0	0
6	1
2	1

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

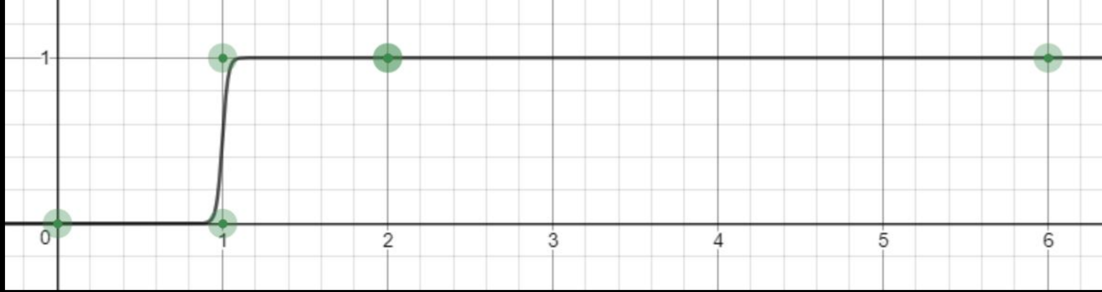
The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.

x	y
2	1
1	0
0	0
6	1
2	1
1	1

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

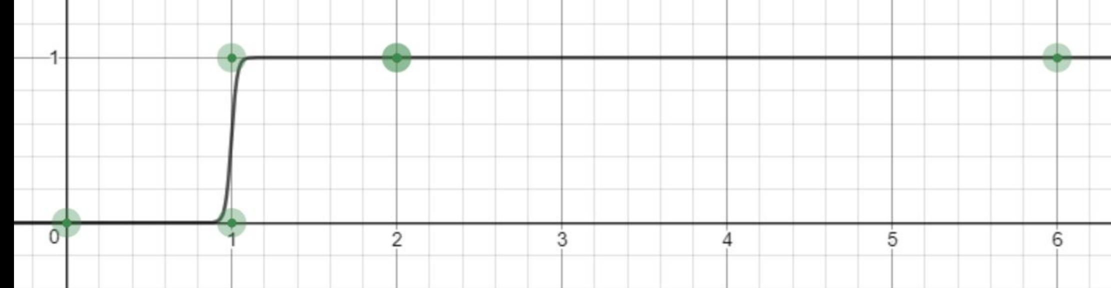
The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.

x	y
2	1
1	0
0	0
6	1
2	1
1	1

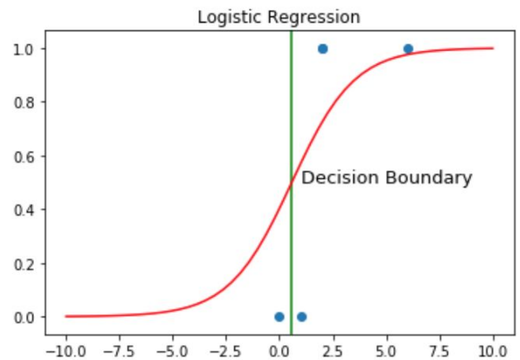
Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;

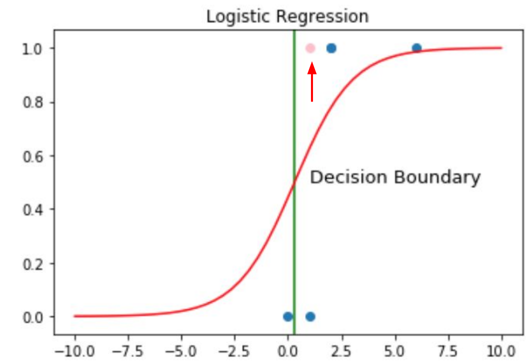
X: number of capital letters in target and surrounding words.

```
Out[43]: [<matplotlib.lines.Line2D at 0x116e68d68>]
```




```
In [78]:  1  -b_0/b_1
Out[78]: 0.5824799517820446

In [28]:  1  logisticRegr.predict(x)
Out[28]: array([1, 1, 0, 1, 1])
```

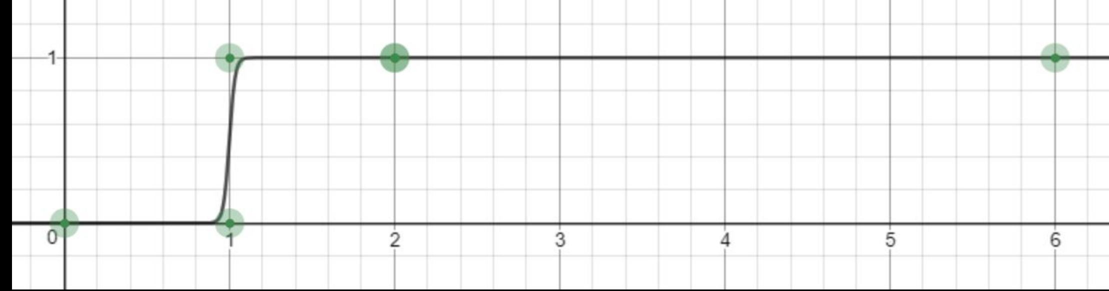
```
Out[80]: [<matplotlib.lines.Line2D at 0x11a60f160>]
```




```
In [81]:  1  -b2_0/b2_1
Out[81]: 0.31089309388058134

In [82]:  1  logisticRegr2.predict(x2)
Out[82]: array([1, 1, 0, 1, 1, 1])
```

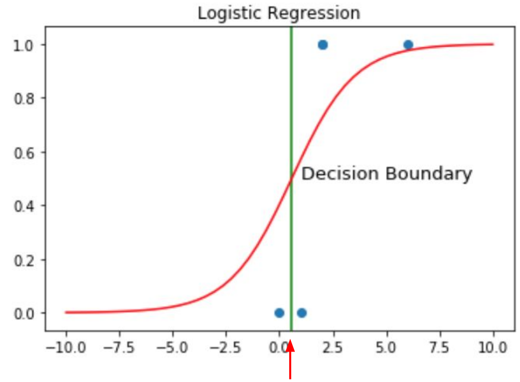
1	1

Logistic Regression



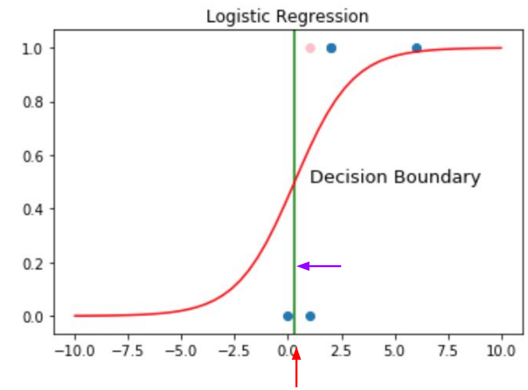
Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

```
Out[43]: [<matplotlib.lines.Line2D at 0x116e68d68>]
```




```
In [78]:  1  -b_0/b_1
Out[78]: 0.5824799517820446

In [28]:  1  logisticRegr.predict(x)
Out[28]: array([1, 1, 0, 1, 1])
```

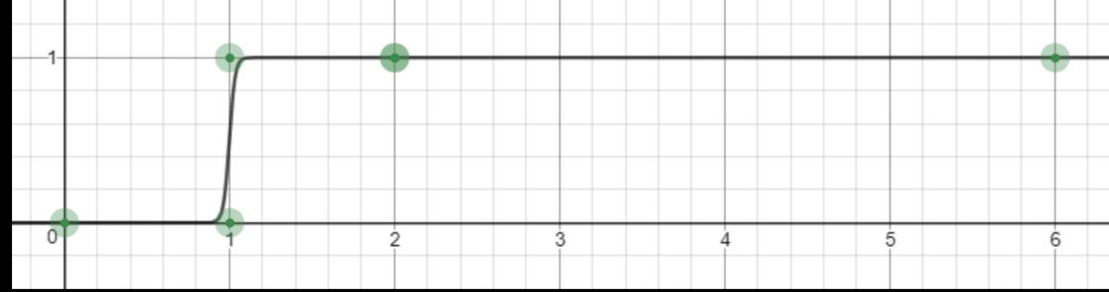
```
Out[80]: [<matplotlib.lines.Line2D at 0x11a60f160>]
```




```
In [81]:  1  -b2_0/b2_1
Out[81]: 0.31089309388058134

In [82]:  1  logisticRegr2.predict(x2)
Out[82]: array([1, 1, 0, 1, 1, 1])
```

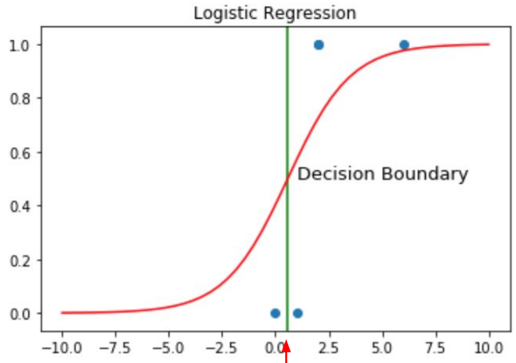
1	1

Logistic Regression



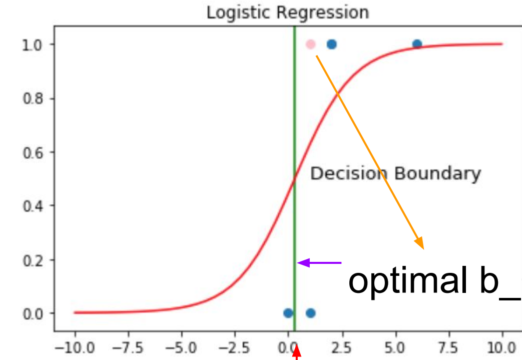
Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

```
Out[43]: [<matplotlib.lines.Line2D at 0x116e68d68>]
```




```
In [78]: -b_0/b_1
Out[78]: 0.5824799517820446

In [28]: logisticRegr.predict(x)
Out[28]: array([1, 1, 0, 1, 1])
```

```
Out[80]: [<matplotlib.lines.Line2D at 0x11a60f160>]
```



optimal b_0, b_1 changed!

```
In [81]: -b2_0/b2_1
Out[81]: 0.31089309388058134

In [82]: logisticRegr2.predict(x2)
Out[82]: array([1, 1, 0, 1, 1, 1])
```

1	1

Logistic Regression on a single feature (*x*)

$Y_i \in \{0, 1\}$; X is a **single value** and can be anything numeric.

$$P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

Logistic Regression on a single feature (*x*)

$Y_i \in \{0, 1\}$; X is a **single value** and can be anything numeric.

$$P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

$$= \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij})}}$$

Logistic Regression on a single feature (*x*)

$Y_i \in \{0, 1\}$; X can be anything numeric.

$$P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

The goal of this function is to: <u>take in the variable *x*</u> and <u>return a probability that *Y* is 1</u>.

Logistic Regression on a single feature (*x*)

Y_i ∈ {0, 1}; X can be anything numeric.

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\boxed{\beta_0}+\boxed{\beta_1}\boxed{x_i}}}{1 + e^{\boxed{\beta_0}+\boxed{\beta_1}\boxed{x_i}}}$$

The goal of this function is to: take in the variable *x* and return a probability that *Y* is 1.

Note that there are only three variables on the right: X_i, B_0, B_1

Logistic Regression on a single feature (*x*)

$Y_i \in \{0, 1\}$; X can be anything numeric.

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\boxed{\beta_0}+\boxed{\beta_1}\boxed{x_i}}}{1 + e^{\boxed{\beta_0}+\boxed{\beta_1}\boxed{x_i}}}$$

The goal of this function is to: take in the variable *x* and return a probability that *Y* is 1.

Note that there are only three variables on the right: X_i, B_0, B_1

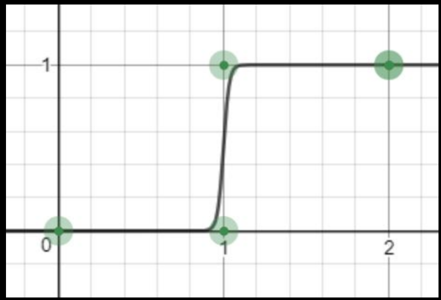
X is given. B_0 and B_1 must be learned.

Logistic Regression on a single feature (*x*)

$Y_i \in \{0, 1\}$; X can be anything numeric.

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\boxed{\beta_0} + \boxed{\beta_1} x_i}}{1 + e^{\boxed{\beta_0} + \boxed{\beta_1} x_i}}$$

HOW? Essentially, try different B_0 and B_1 values until "best fit" to the training data (example X and Y).



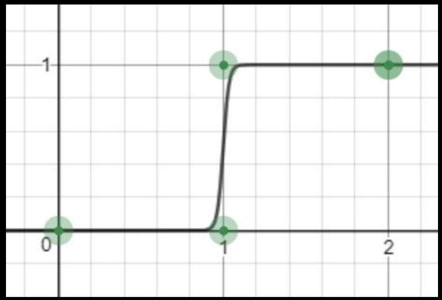
X is given. B_0 and B_1 must be **learned**.

“best fit” : whatever maximizes the likelihood function:

$$L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\boxed{\beta_0}+\boxed{\beta_1}x_i}}{1 + e^{\boxed{\beta_0}+\boxed{\beta_1}x_i}}$$

HOW? Essentially, try different B_0 and B_1 values until “best fit” to the training data (example X and Y).



X is given. B_0 and B_1 must be **learned.**

“best fit” : whatever maximizes the likelihood function:

$$\boxed{L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

“best fit” : whatever maximizes the likelihood function:

$$\boxed{L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

“best fit” : whatever maximizes the *likelihood* function:

$$\boxed{L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

“best fit” : more efficient to maximize *log likelihood* :

“best fit” : whatever maximizes the *likelihood* function:

$$\boxed{L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

“best fit” : more efficient to maximize *log likelihood* :

$$\ell(\beta) = \sum_{i=1}^{N} y_i log\, p(x_i) + (1 - y_i) log\, (1 - p(x_i))$$

“best fit” : whatever maximizes the *likelihood* function:

$$\boxed{L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

“best fit” : more efficient to maximize *log likelihood* :

$$\ell(\beta) = \sum_{i=1}^{N} y_i log\, p(x_i) + (1 - y_i) log\, (1 - p(x_i))$$

“best fit” for neural networks: software designed to **minimize** rather than maximize (typically, normalized by N, the number of examples.)

“best fit” : whatever maximizes the *likelihood* function:

$$\boxed{L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

“best fit” : more efficient to maximize *log likelihood* :

$$\ell(\beta) = \sum_{i=1}^{N} y_i log\, p(x_i) + (1 - y_i) log\, (1 - p(x_i))$$

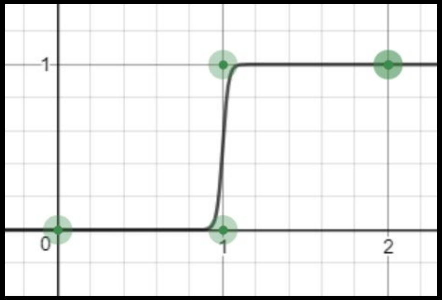
“best fit” for neural networks: software designed to **minimize** rather than maximize (typically, normalized by N, number of examples.) "*log loss*" or *"normalized log loss":*

$$\boxed{J(\beta) = -\frac{1}{N} \sum_{i=1}^{N} y_i log\, p(x_i) + (1 - y_i) log\, (1 - p)(x_i))}$$

X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding: integer
- Begins with capital letter: {0, 1}
- Preceded by “the”? {0, 1}



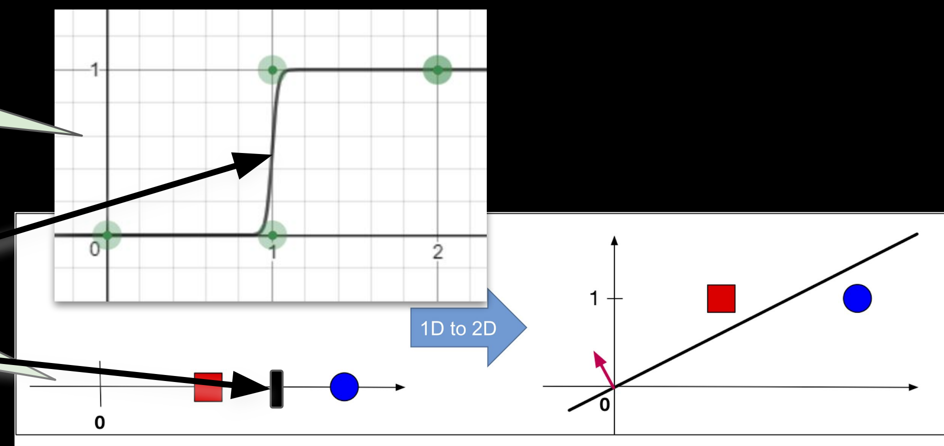
X can be multiple features

Often we want to make a classification based on multiple features:

- Numb
  surroundi
- Begins wit {0, 1}
- Pred

Y-axis is Y (i.e. 1 or 0)

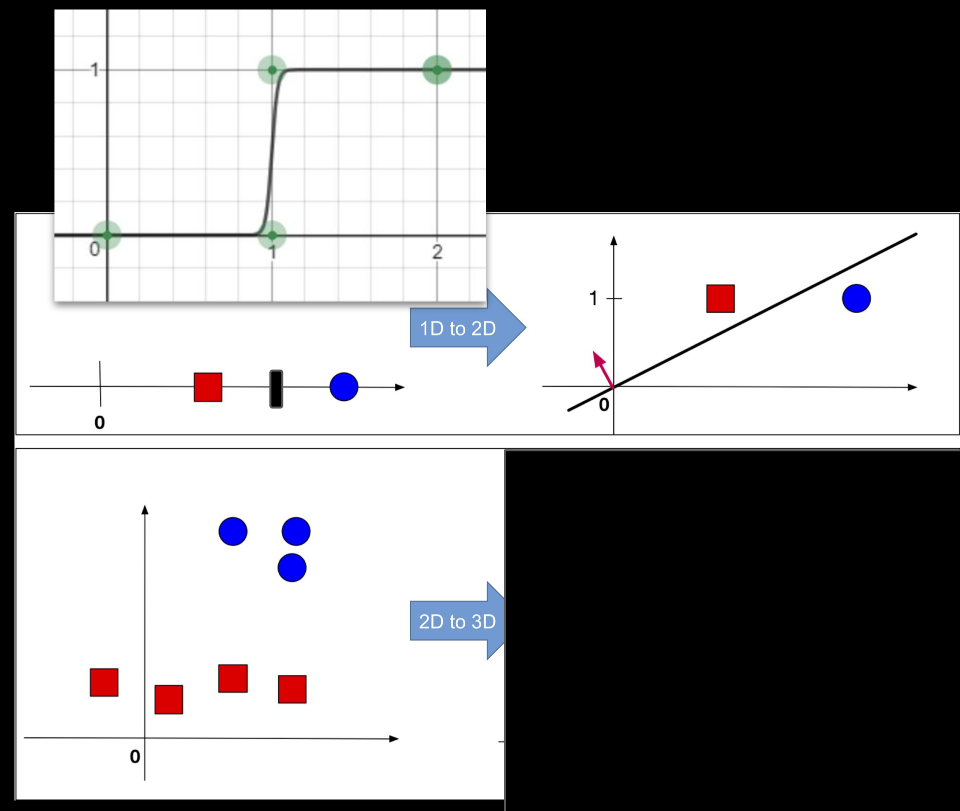
To make room for multiple Xs, let's get rid of y-axis. Instead, show **decision point**.



X can be multiple features

Often we want to make a classification based on multiple features:

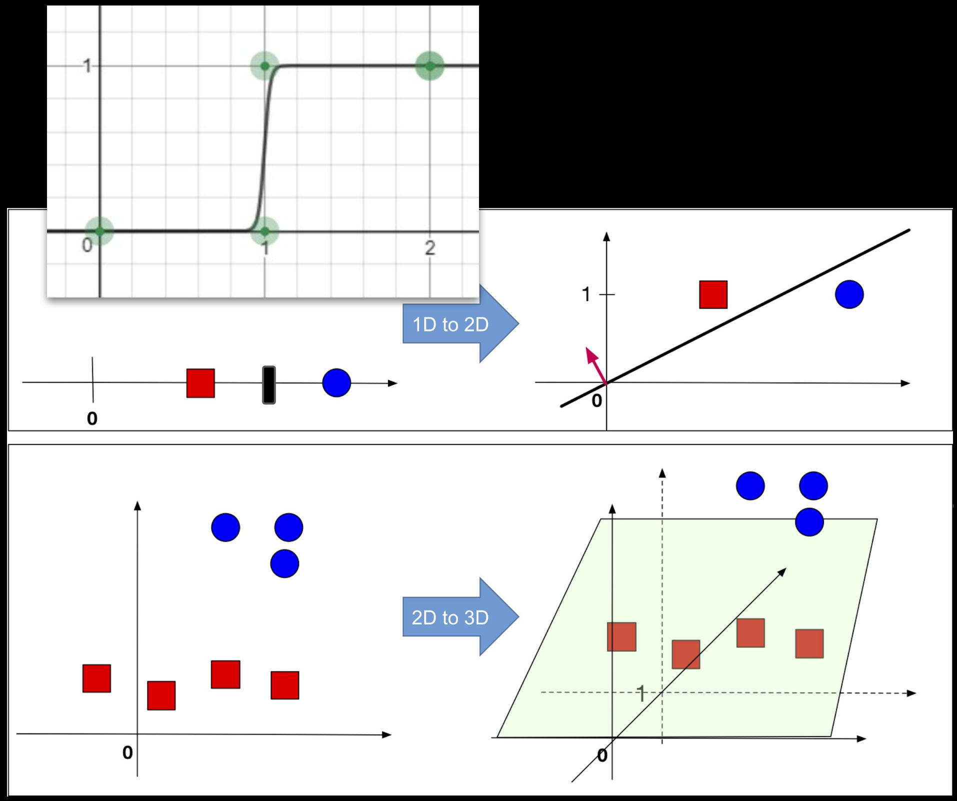
- Number of capital letters surrounding: integer
- Begins with capital letter: {0, 1}
- Preceded by “the”? {0, 1}



X can be multiple features

Often we want to make a classification based on multiple features:

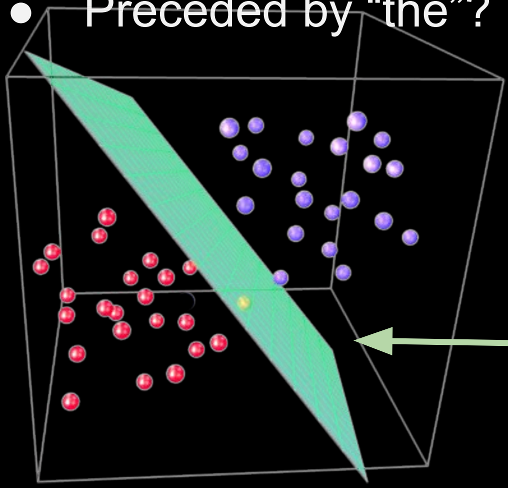
- Number of capital letters surrounding: integer
- Begins with capital letter: {0, 1}
- Preceded by “the”? {0, 1}



X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding: integer
- Begins with capital letter: {0, 1}
- Preceded by “the”? {0, 1}



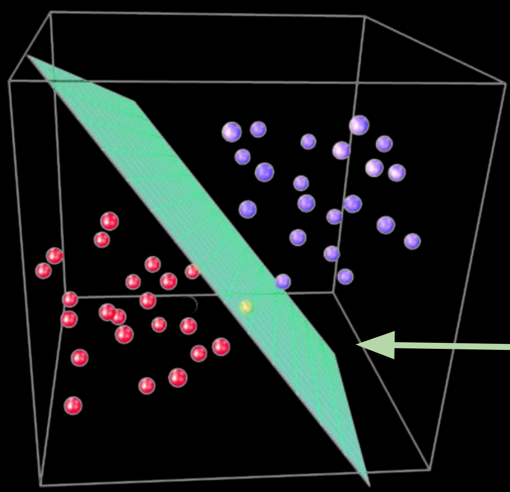
We’re learning a linear (i.e. flat) *separating hyperplane*, but fitting it to a *logit* outcome.

(https://www.linkedin.com/pulse/predicting-outcomes-probabilities-logistic-regression-konstantinidis/)

Logistic Regression

$Y_i \in \{0, 1\}$; X can be anything numeric.

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}$$



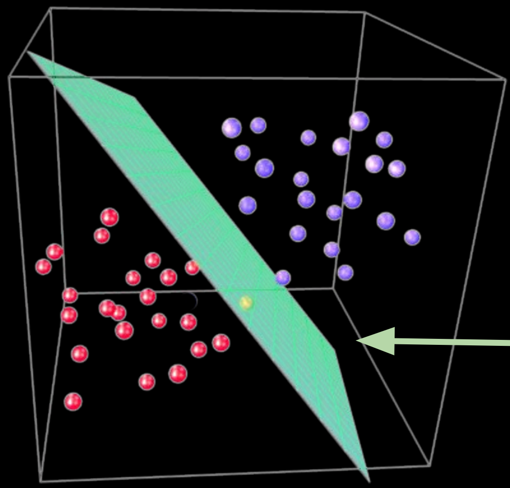
We're still learning a linear *separating hyperplane*, but fitting it to a *logit* outcome.

(https://www.linkedin.com/pulse/predicting-outcomes-probabilities-logistic-regression-konstantinidis/)

Logistic Regression

$Y_i \in \{0, 1\}$; X can be anything numeric.

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}$$

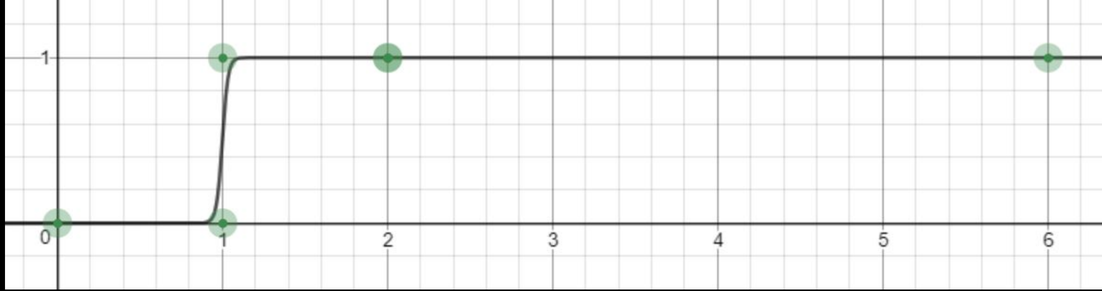


$$logit(p_i) = log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \sum_{j=1}^{m} \boxed{\beta_j x_{ij}} = 0$$

We're still learning a linear *separating hyperplane*, but fitting it to a *logit* outcome.

(https://www.linkedin.com/pulse/predicting-outcomes-probabilities-logistic-regression-konstantinidis/)

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;
X: number of capital letters in target and surrounding words.

They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

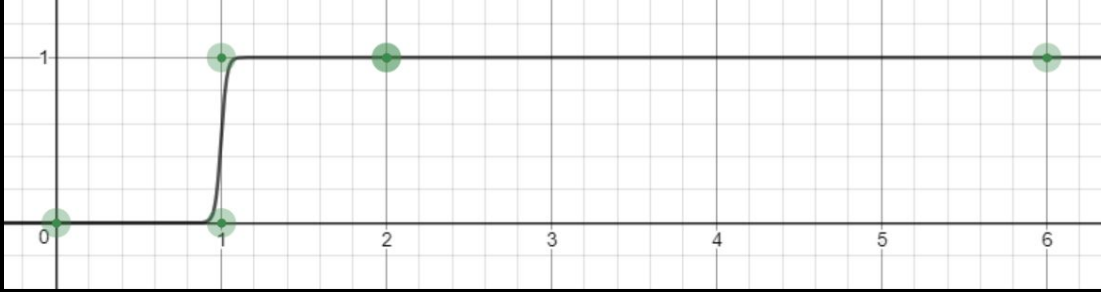
The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.

x	y
2	1
1	0
0	0
6	1
2	1
1	1

Logistic Regression



Example: Y: 1 if target is a part of a proper noun, 0 otherwise;

X1: number of capital letters in target and surrounding words.

Let's add a feature! X2: does the target word start with a capital letter?

They attend Stony Brook University. *Next to the brook Gandalf lay thinking.*

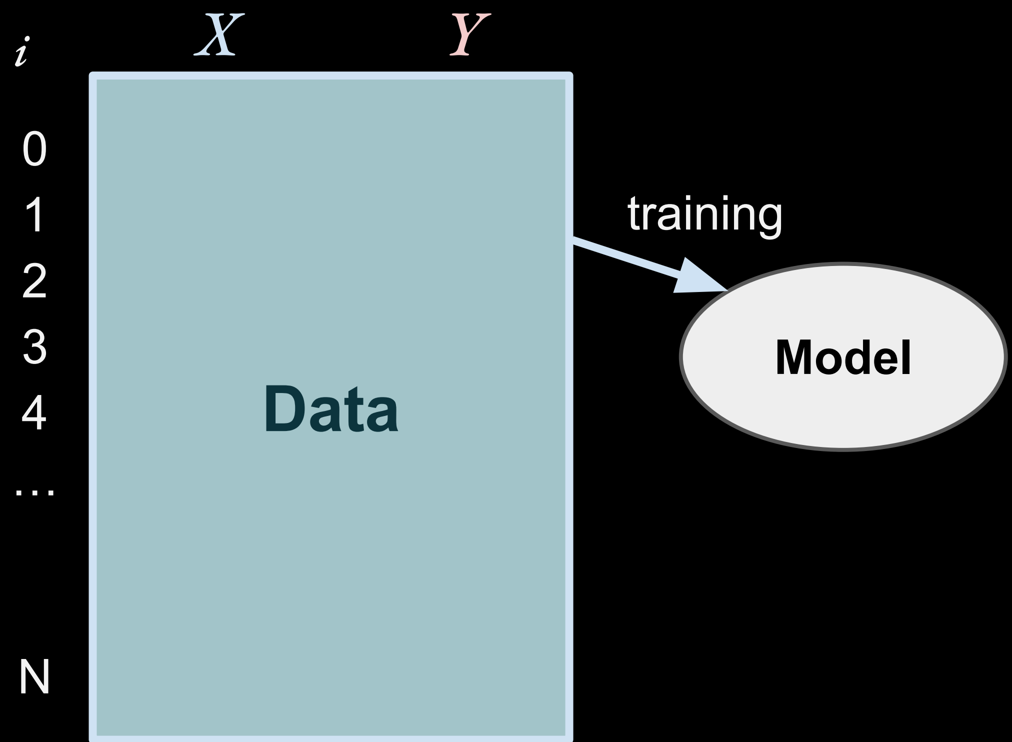
The trail was very stony. *Her degree is from SUNY Stony Brook.*

The Taylor Series was first described by Brook Taylor, the mathematician.

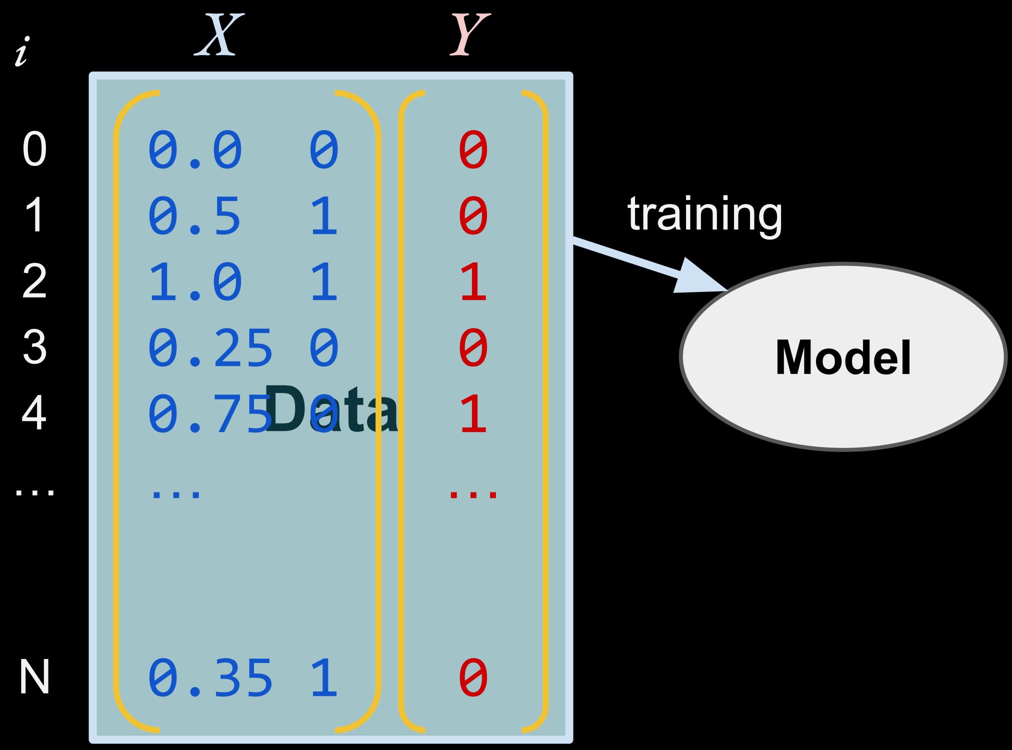
They attend Binghamton.

x2	x1	y
1	2	1
0	1	0
0	0	0
1	6	1
1	2	1
1	1	1

Machine Learning: How to setup data



Machine Learning: How to setup data



Machine Learning: How to setup data

"Corpus"

raw data: sequences of characters

i	X		Y
0	0.0	0	0
1	0.5	1	0
2	1.0	1	1
3	0.25	0	0
4	0.75	0	1
…	…		…
N	0.35	1	0

Data

training

Machine Learning: How to setup data

Feature Extraction

"Corpus"

raw data: sequences of characters

--pull out *observations* and *feature vector* per observation.

i	X		Y
0	0.0	0	0
1	0.5	1	0
2	1.0	1	1
3	0.25	0	0
4	0.75	0	1
…	…		…
N	0.35	1	0

Data

training

Machine Learning: How to setup data

"Corpus"

raw data: sequences of characters

Feature Extraction

--pull out *observations* and *feature vector* per observation.

e.g.: words, sentences, documents, users.

i	X		Y
0	0.0	0	0
1	0.5	1	0
2	1.0	1	1
3	0.25	0	0
4	0.75	0	1
…	…		…
N	0.35	1	0

Data

training

Machine Learning: How to setup data

Feature Extraction

"Corpus"

raw data:
sequences of
characters

--pull out *observations* and *feature vector* per observation.

e.g.: words, sentences, documents, users.

row of features; e.g.

→ *number of capital letters*

→ *whether "I" was mentioned or not*

i	X		Y
0	0.0	0	0
1	0.5	1	0
2	1.0	1	1
3	0.25	0	0
4	0.75	0	1
…	…		…
N	0.35	1	0

Data

training

Machine Learning: How to setup data

Feature Extraction

"Corpus"

raw data: sequences of characters

--pull out *observations* and *feature vector* per observation.

e.g.: words, sentences, documents, users.

row of features; e.g.

→ *number of capital letters*

→ *whether "I" was mentioned or not*

→ *k features indicating whether k words were mentioned or not*

i	X		Y
0	0.0	0	0
1	0.5	1	0
2	1.0	1	1
3	0.25	0	0
4	0.75	0	1
…	…		…
N	0.35	1	0

Data

training

Machine Learning: How to setup data

Feature Extraction

X Y

Multi-hot Encoding

- Each word gets an index in the vector
- 1 if present; 0 if not

"Corpus"

raw data:
sequences of characters

of features; e.g.

→ *number of capital letters*

→ *whether "I" was mentioned or not*

→ *k features indicating whether k words were mentioned or not*

Data

Machine Learning: How to setup data

Feature Extraction

X Y

Multi-hot Encoding

- Each word gets an index in the vector
- 1 if present; 0 if not

Feature example: is word present in document?

The book was interesting so I was happy .

"Corpus"

raw data: sequences of characters

of features; e.g.

→ number of capital letters

→ whether "I" was mentioned or not

→ k features indicating whether k words were mentioned or not

Data

Machine Learning: How to setup data

Feature Extraction

X Y

Multi-hot Encoding

- Each word gets an index in the vector
- 1 if present; 0 if not

Feature example: is word present in document?

The book was interesting so I was happy .

```
[0, 1, 1, 0, 1, …, 1, 0, 1, 1, 0, 1, …, 1]
```
^{k}

→ *k features indicating whether k words were mentioned or not*

"Corpus"

raw data: sequences of characters

Data

Machine Learning: How to setup data

Feature Extraction

X Y

Multi-hot Encoding

- Each word gets an index in the vector
- 1 if present; 0 if not

Feature example: is word present in document

“Corpus”

raw data: sequences of characters

Data

The book was interesting so I was happy .

`[0, 1, 1, 0, 1, …, 1, 0, 1, 1, 0, 1, …, 1]`^{k}

a

→ *k features indicating whether k words were mentioned or not*

sad

Machine Learning: How to setup data

Feature Extraction

X Y

Multi-hot Encoding

- Each word gets an index in the vector
- 1 if present; 0 if not

Feature example: is previous word "the"?

The ***book*** *was interesting so I was happy .*

`[0, 1, 1, 0, 1, …, 1, 0, 1, 1, 0, 1, …, 1]`^{k}

→ *k features indicating whether k words were mentioned or not*

"Corpus"

raw data: sequences of characters

Data

Machine Learning: How to setup data

Feature Extraction

X Y

<u>Multi-hot Encoding</u>

- Each word gets an index in the vector
- 1 if present; 0 if not

Feature example: is previous word “the”?

“Corpus”

raw data: sequences of characters

The ***book*** *was interesting so I was happy .*

~~[0, 1, 1, 0, 1, …, 1, 0, 1, 1, 0, 1, …, 1]~~^{k}

→ *k features indicating whether k words were mentioned or not*

Data

Machine Learning: How to setup data

Feature Extraction

X Y

“Corpus”

raw data: sequences of characters

Data

<u>One-hot Encoding</u>

- Each word gets an index in the vector
- All indices 0 except present word:

  Feature example: is previous word “the”?

The ***book*** *was interesting so I was happy .*

`[0, 1, 0, 0, 0, …, 0, 0, 0, 0, 0, 0, …, 0]`^{k}

→ *k features indicating whether k words were mentioned or not*

Machine Learning: How to setup data

Feature Extraction

X Y

One-hot Encoding

- Each word gets an index in the vector
- All indices 0 except present word:

Feature example: which is previous word?

The book was interesting so I was happy .

[0, 1, 0, 0, 0, ..., 0, 0, 0, 0, 0, 0, ..., 0]^k

[0, 0, 1, 0, 0, ..., 0, 0, 0, 0, 0, 0, ..., 0]^k

"Corpus"

raw data: sequences of characters

Data

Machine Learning: How to setup data

Feature Extraction

X Y

Data

“Corpus”

raw data: sequences of characters

One-hot Encoding

- Each word gets an index in the vector
- All indices 0 except present word:

  Feature example: which is previous word?

The book was interesting so I was happy .

[0, 1, 0, 0, 0, …, 0, 0, 0, 0, 0, 0, …, 0]^k

[0, 0, 1, 0, 0, …, 0, 0, 0, 0, 0, 0, …, 0]^k

Machine Learning: How to setup data

Feature Extraction

X Y

Multiple One-hot encodings for one observation

(1) word before; (2) word after

*The book was **interesting** so I was happy .*

`[0, 0, 0, 0, 1, 0, …, 0]`^{k} `[0, …, 0, 1, 0, …, 0]`^{k}

Machine Learning: How to setup data

Feature Extraction

X Y

<u>Multiple One-hot encodings for one observation</u>

(1) word before; (2) word after

"Corpus"

*The book was **interesting** so I was happy .*

raw data: sequences of characters

Data

$[0, 0, 0, 0, 1, 0, \ldots, 0]^k$ $[0, \ldots, 0, 1, 0, \ldots, 0]^k$

=

$[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0]^{2k}$

Machine Learning: How to setup data

Feature Extraction

X Y

Multiple One-hot encodings for one observation

(1) word before; (2) word after; (3) percent capitals

The book was ***Interesting*** *so I was happy .*

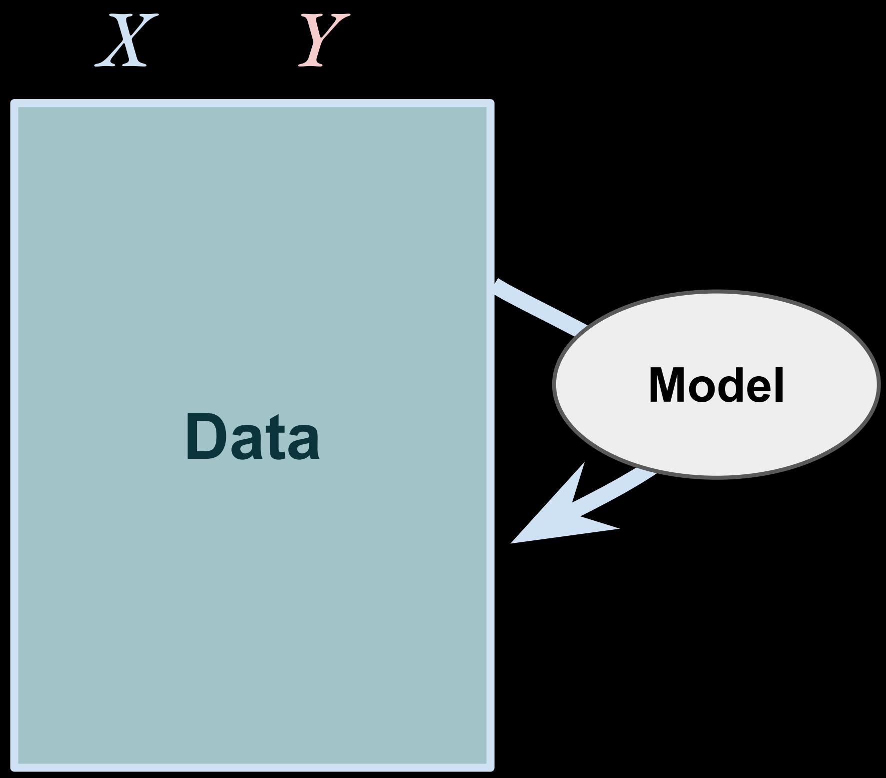
$[0, 0, 0, 0, 1, 0, \ldots, 0]^k \; [0, \ldots, 0, 1, 0, \ldots, 0]^k$

$=$

$[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0]^{2k}$

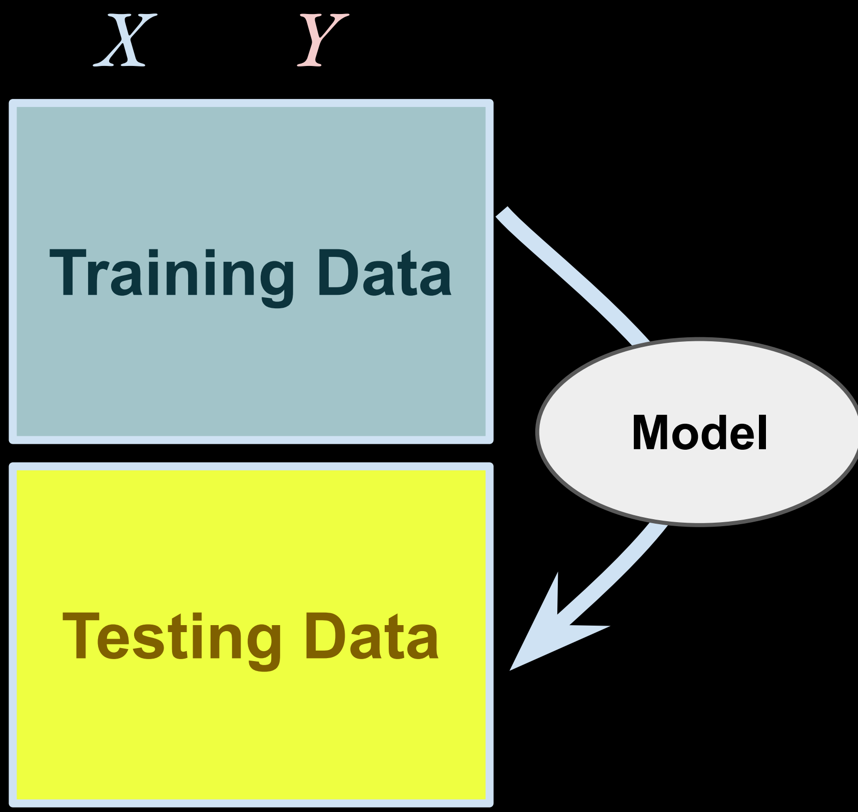
$[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0, 0.09]^{2k+1}$

Machine Learning: How to setup data



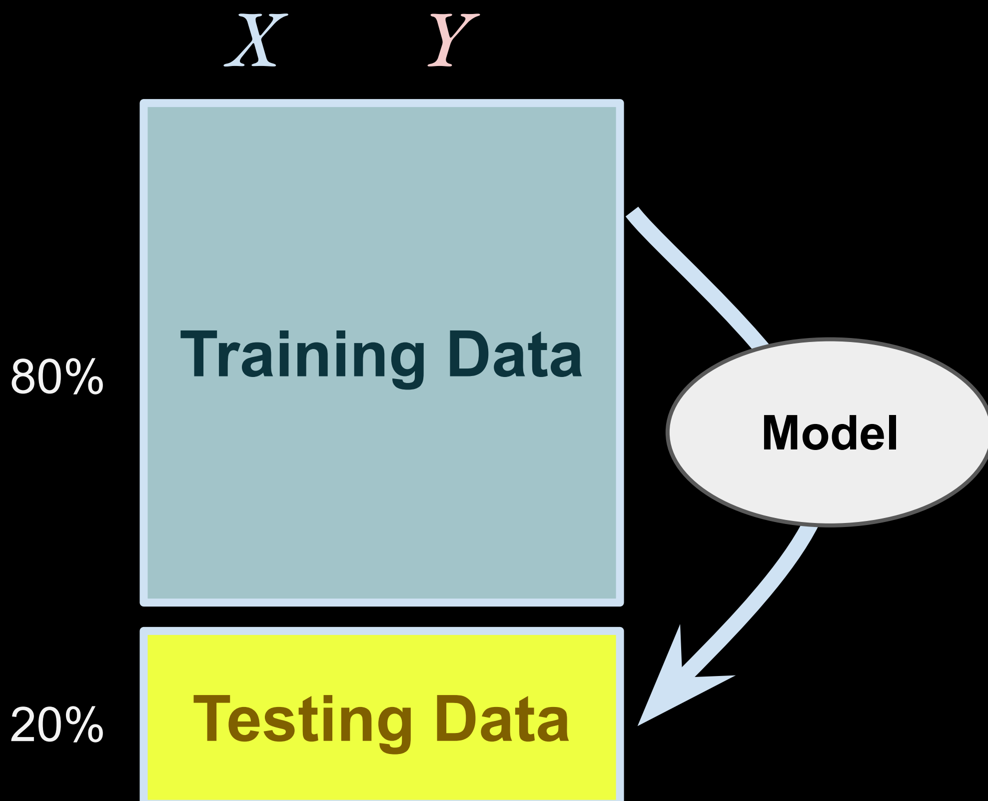
Does the
model hold up?

Machine Learning Goal: Generalize to new data



Does the model hold up?

Machine Learning Goal: Generalize to new data



Does the model hold up?

Logistic Regression - Regularization

$$X \qquad = \qquad Y$$

0.5	0	0.6	1	0	0.25	1
0	0.5	0.3	0	0	0	1
0	0	1	1	1	0.5	0
0	0	0	0	1	1	0
0.25	1	1.25	1	0.1	2	1

Logistic Regression - Regularization

$$X = Y$$

0.5	0	0.6	1	0	0.25	1
0	0.5	0.3	0	0	0	1
0	0	1	1	1	0.5	0
0	0	0	0	1	1	0
0.25	1	1.25	1	0.1	2	1

Logistic Regression - Regularization

$$X = Y$$

x_1	x_2	...				Y
0.5	0	0.6	1	0	0.25	1
0	0.5	0.3	0	0	0	1
0	0	1	1	1	0.5	0
0	0	0	0	1	1	0
0.25	1	1.25	1	0.1	2	1

$$1.2 + -63*x_1 + 179*x_2 + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

Logistic Regression - Regularization

$$X = Y$$

x_1	x_2	...				Y
0.5	0	0.6	1	0	0.25	1
0	0.5	0.3	0	0	0	1
0	0	1	1	1	0.5	0
0	0	0	0	1	1	0
0.25	1	1.25	1	0.1	2	1

$$1.2 + -63*x_1 + \boxed{179*x_2} + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

Logistic Regression - Regularization

X = Y

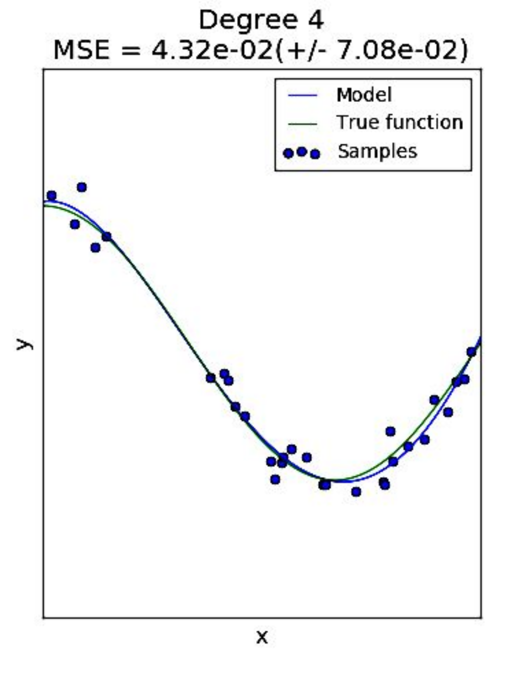
x_1	x_2	...				Y
0.5	0	0.6	1	0	0.25	1
0	0.5	[illegible]	[illegible]	[illegible]	0	1
0	0	[illegible]	[illegible]	[illegible]	0.5	0
0	0	0	0	1	1	0
0.25	1	1.25	1	0.1	2	1

"overfitting"

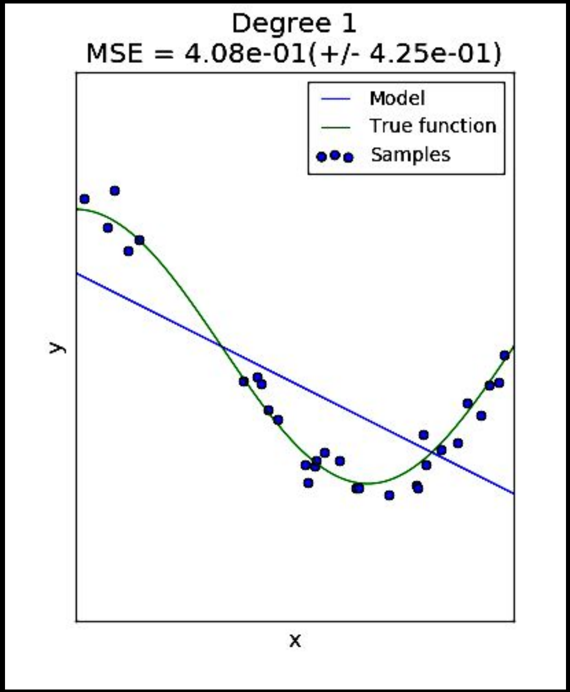
$$1.2 + -63*x_1 + \boxed{179*x_2} + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

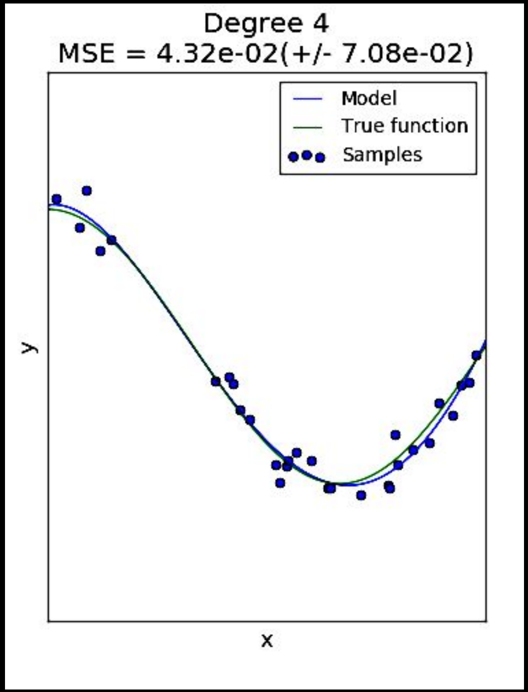
Python Example

Overfitting (1-d non-linear example)



Overfitting (1-d non-linear example)

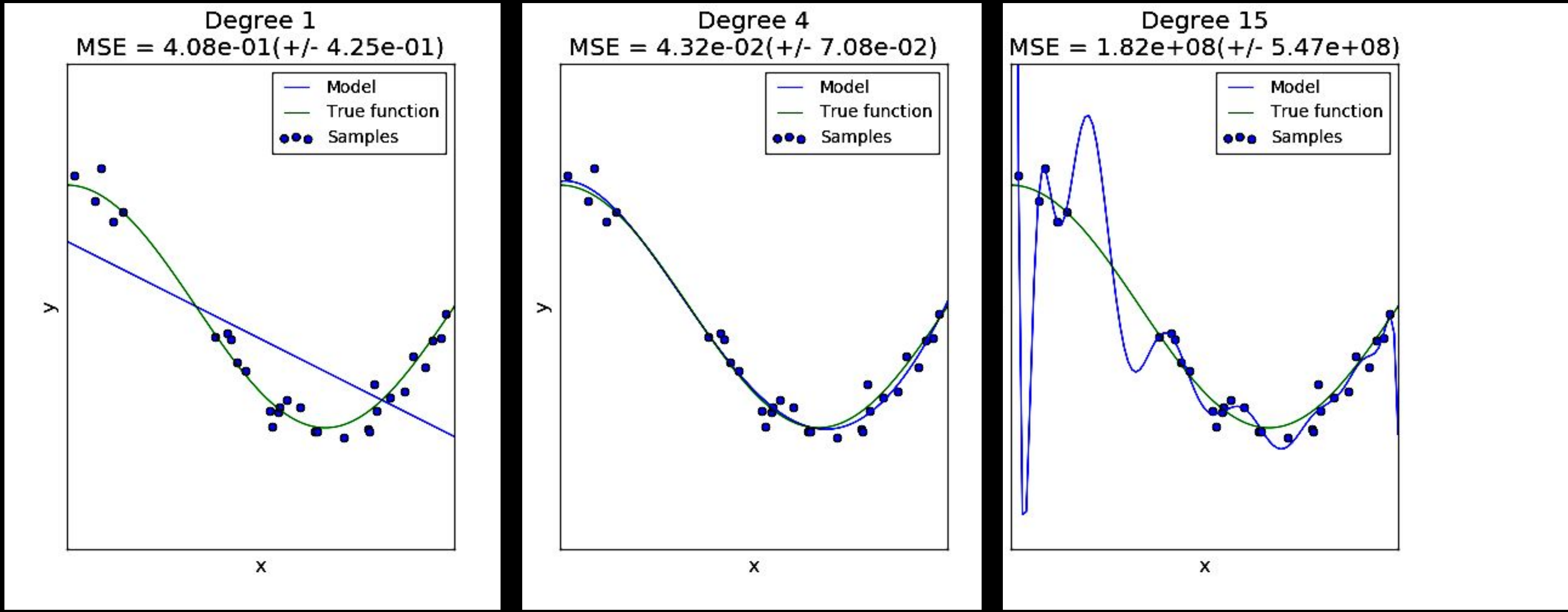




Underfit

(image credit: Scikit-learn; in practice data are rarely this clear)

Overfitting (1-d non-linear example)



Underfit

Overfit

(image credit: Scikit-learn; in practice data are rarely this clear)

Logistic Regression - Regularization

x_1	x_2	…	X			$=$	Y
0.5	0	0.6	1	0	0.25		1
0	0.5	[illegible]	[illegible]	[illegible]	0		1
0	0	[illegible]	[illegible]	[illegible]	0.5		0
0	0	0	0	1	1		0
0.25	1	1.25	1	0.1	2		1

"overfitting"

$$1.2 + -63*x_1 + \boxed{179*x_2} + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

Logistic Regression - Regularization

X = Y

x_1	x_2	...				Y
0.5	0	0.6			[illegible]	1
0	0.5					1
0						0
0						0
0.25				0.1	2	1

"overfitting": generally due to trying to fit too many features given the number of observations.

$$1.2 + -63*x_1 + 17[illegible]*x_2 + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

Logistic Regression - Regularization

$$X \qquad\qquad = \quad Y$$

x_1	x_2	Y
0.5	0	1
0	0.5	1
0	0	0
0	0	0
0.25	1	1

What if only 2 predictors?

Logistic Regression - Regularization

$$X = Y$$

x_1	x_2	Y
0.5	0	1
0	0.5	1
0	0	0
0	0	0
0.25	1	1

What if only 2 predictors?

A: better fit

$$0 + 2*x_1 + 2*x_2 = logit(Y)$$

Logistic Regression - Regularization

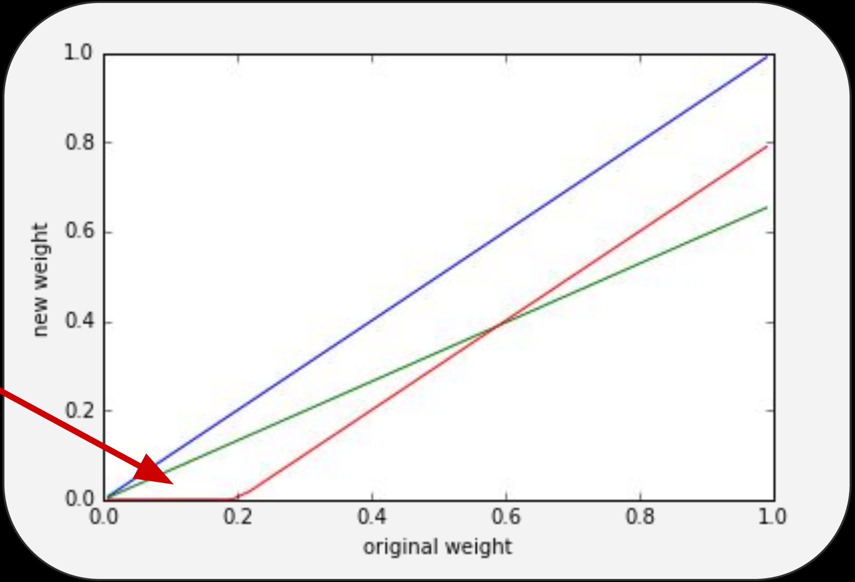
L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

Logistic Regression - Regularization

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.



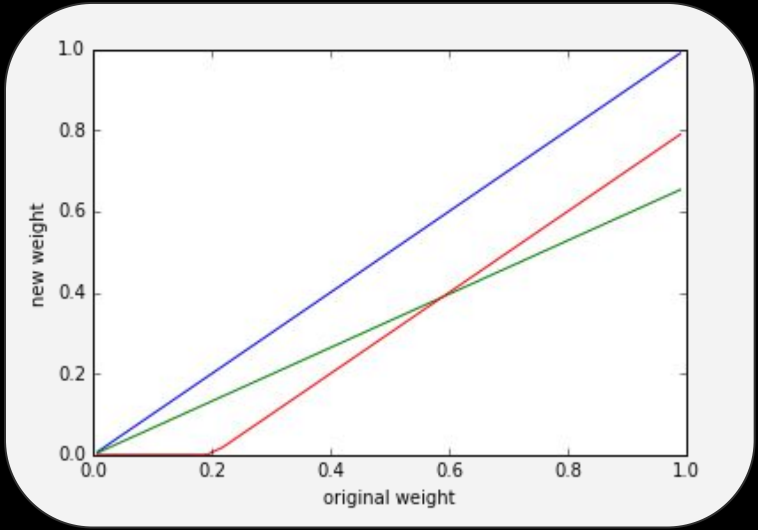
Logistic Regression - Regularization

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

$$L(\beta_0, \beta_1, ..., \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

set betas that maximize L



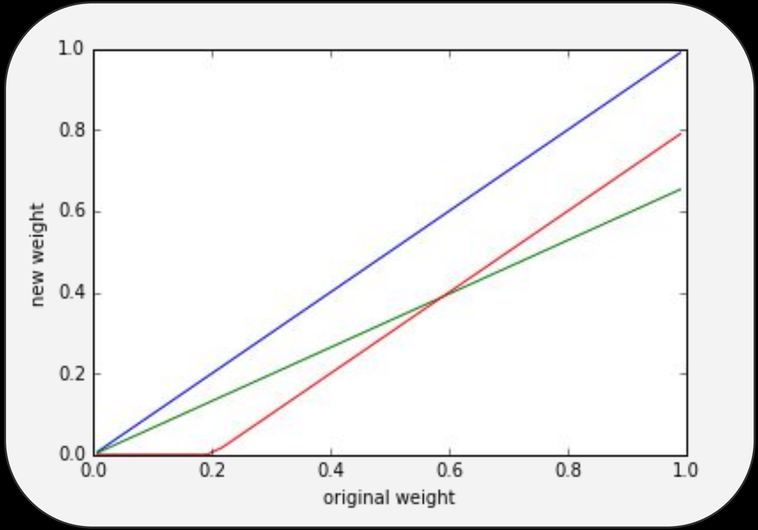
Logistic Regression - Regularization

L1 Regularization - "The Lasso"

Zeros out features by adding values that keep from perfectly fitting the data.

$$L(\beta_0, \beta_1, ..., \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} - \frac{1}{C}\sum_{j=1}^{m} |\beta_j|$$

set betas that maximize *penalized* L



Logistic Regression - Regularization

Sometimes written as:

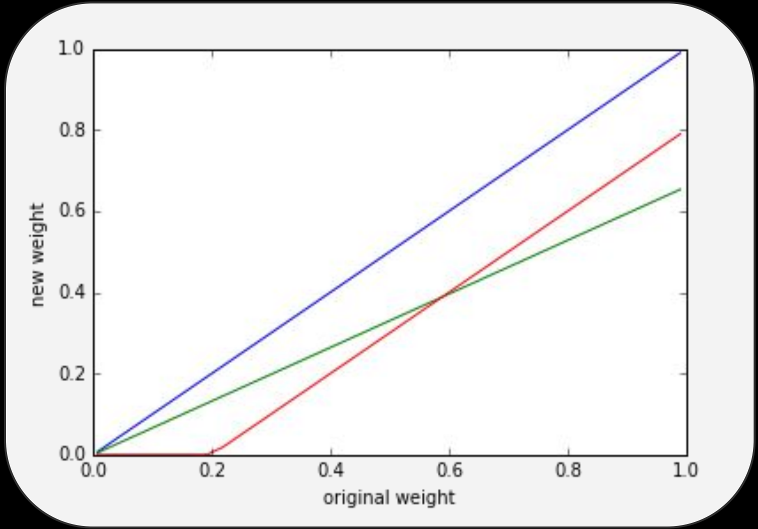
$||\beta||_1$

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

$$L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} - \frac{1}{C}\overbrace{\sum_{j=1}^{m} |\beta_j|}$$

set betas that maximize *penalized* L



Logistic Regression - Regularization

Sometimes written as:

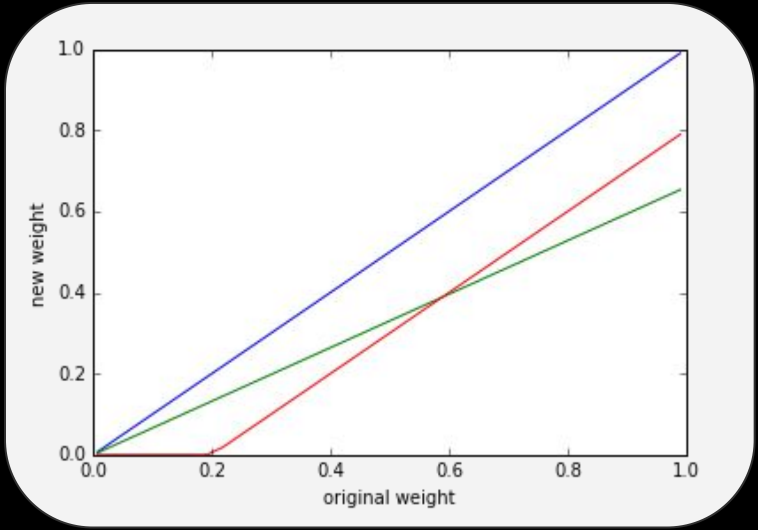
$$||\beta_j||_2^2$$

L2 Regularization - "Ridge"

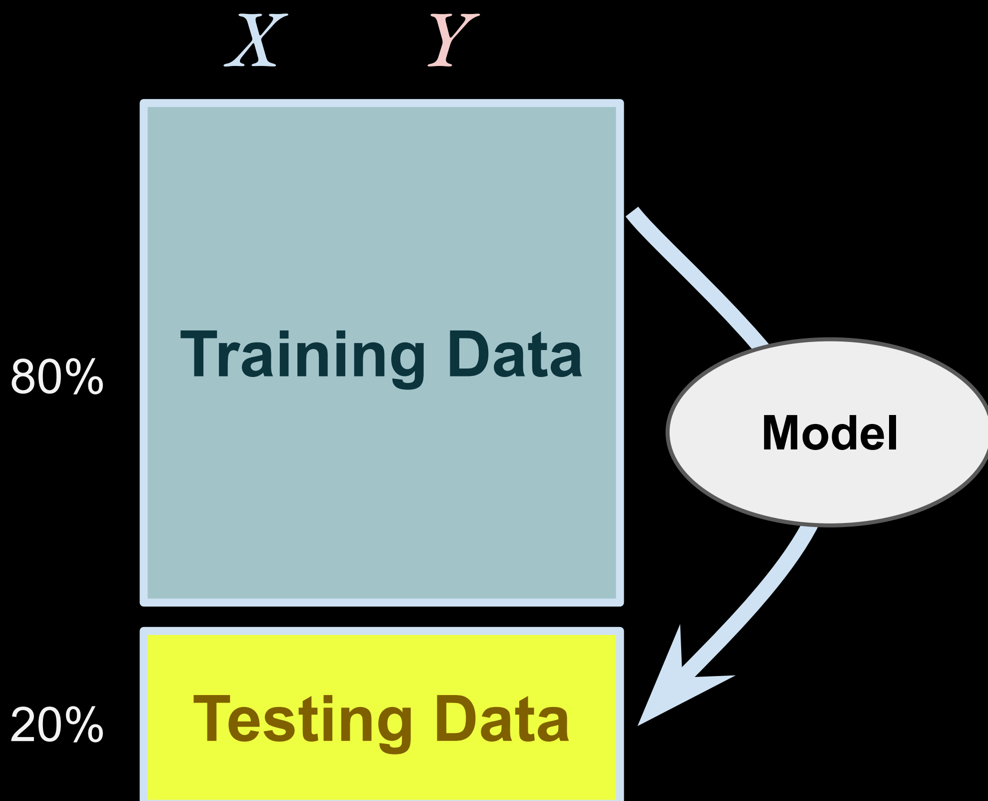
Shrinks features by adding values that keep from perfectly fitting the data.

$$L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} - \frac{1}{C}\sum_{j=1}^{m}\beta_j^2$$

set betas that maximize *penalized* L

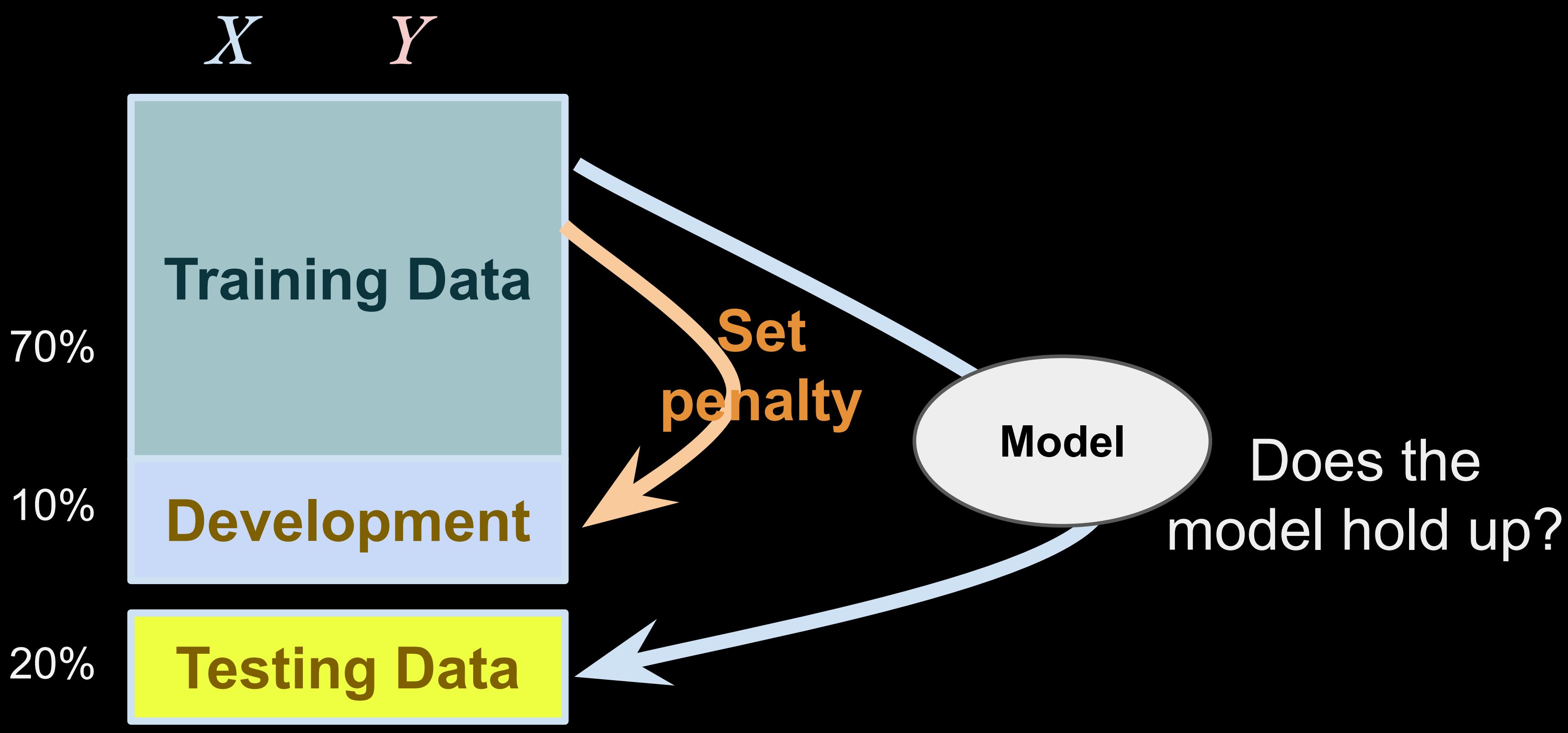


Machine Learning Goal: Generalize to new data



Does the model hold up?

Machine Learning Goal: Generalize to new data



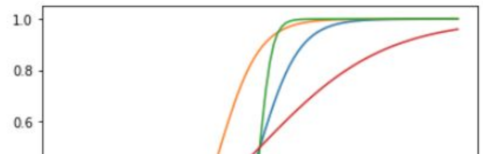
Example

See notebook on website.

In [44]:
```
%matplotlib inline

#above allows plots to discplay on the screen.

#python includes
import sys

#standard probability includes:
import numpy as np #matrices and data structures
import scipy.stats as ss #standard statistical operations
import pandas as pd #keeps data organized, works well with data
import matplotlib
import matplotlib.pyplot as plt #plot visualization
```

In [53]:
```
#let's just look at what happens to the logit function as we change the beta coefficients

def logistic_function(x):
    return np.exp(x) / (1+np.exp(x))

def logistic_function_with_betas(x, beta0=0, beta1=1):
    #now using linear function: beta0 + beta1*x for the exponent:
    return np.exp(beta0 + beta1*x) / (1+np.exp(beta0 + beta1*x))

xpoints = np.linspace(-10, 10, 100)
plt.plot(xpoints, [logistic_function(x) for x in xpoints])
plt.plot(xpoints, [logistic_function_with_betas(x, 2, 1) for x in xpoints]) #shifts the intercept with zero
plt.plot(xpoints, [logistic_function_with_betas(x, 0, 3.145914159653) for x in xpoints])#twists the line verically
plt.plot(xpoints, [logistic_function_with_betas(x, 0, 1/3.145914159653) for x in xpoints]) #twists it horizontally
```

Out[53]: [<matplotlib.lines.Line2D at 0x2691f435f60>]



For 2021: add multinomial

Logistic Regression - Review

- Probabilistic Classification: $P(Y \mid X)$
- Learn logistic curve based on example data
  - training + development + testing data
- Set betas based on maximizing the *likelihood* (or based on minimizing *log loss*)
  - “shifts” and “twists” the logistic curve
  - separation represented by hyperplane at 0.50
- Multivariate features: One-hot encodings
- Overfitting and Regularization

Extra Material

Alternative to gradient descent:

$$L(\beta_0, \beta_1, ..., \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\boxed{\beta_0}+\boxed{\beta_1}x_i}}{1 + e^{\boxed{\beta_0}+\boxed{\beta_1}x_i}}$$

To estimate β, one can use *reweighted least squares:*

(Wasserman, 2005; Li, 2010)

set $\hat{\beta}_0 = ... = \hat{\beta}_m = 0$ (remember to include an intercept)

1. Calculate p_i and let W be a diagonal matrix where element$(i, i) = p_i(1 - p_i)$.
2. Set $z_i = logit(p_i) + \frac{Y_i - p_i}{p_i(1 - p_i)} = X\hat{\beta} + \frac{Y_i - p_i}{p_i(1 - p_i)}$
3. Set $\hat{\beta} = (X^T W X)^{-1} X^T W z$ //weighted lin. reg. of Z on Y.
4. Repeat from 1 until $\hat{\beta}$ converges.

Alternative to gradient descent:

$$L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

This is just one way of finding the betas that maximize the likelihood function. In practice, we will use existing libraries that are fast and support additional useful steps like **regularization**..

To estimate β, one can use *reweighted least squares:*

(Wasserman, 2005; Li, 2010)

set $\hat{\beta}_0 = \ldots = \hat{\beta}_m = 0$ (remember to include an intercept)

1. Calculate p_i and let W be a diagonal matrix where element$(i, i) = p_i(1 - p_i)$.
2. Set $z_i = logit(p_i) + \frac{Y_i - p_i}{p_i(1 - p_i)} = X\hat{\beta} + \frac{Y_i - p_i}{p_i(1 - p_i)}$
3. Set $\hat{\beta} = (X^T W X)^{-1} X^T W z$ //weighted lin. reg. of Z on Y.
4. Repeat from 1 until $\hat{\beta}$ converges.