NYU FRE 7773 - Week 4

Machine Learning in Financial Engineering

Ethan Rosenthal

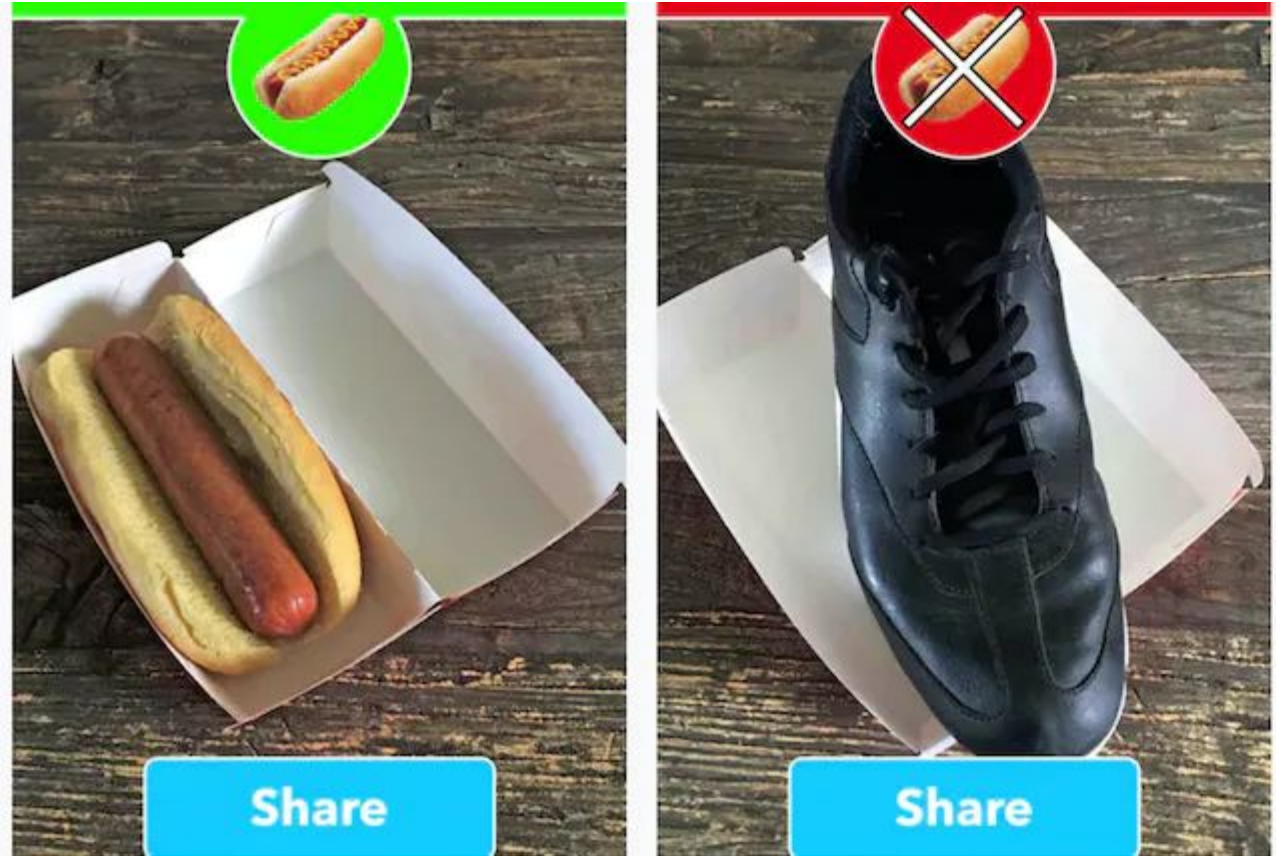
Linear Models for Classification

Machine Learning in Financial Engineering

Ethan Rosenthal

Classification


Announcing
$0
Online Stock, ETF, and Options Commissions
+ A Satisfaction Guarantee
If you're not happy for any reason, we'll refund your eligible fee and work with you to make things right.
GET THE DETAILS
charles SCHWAB
Own your tomorrow
ENGLISH
ESPAÑOL
中文
Log In
Thursday, October 17, 2019
The New York Times
Today's Paper



https://www.theverge.com/2017/6/26/15876006/hot-dog-app-android-silicon-valley https://huggingface.co/EleutherAI/gpt-j-6B

Logistic Regression

The ML Recipe

1. Think up some model
2. Feed **data** into the model and make predictions.
3. Calculate the loss between predictions and true values.
4. Determine the model parameters that produce the minimum loss.

The ML Recipe

1. Think up some model
2. Feed **data** into the model and make predictions.
3. Calculate the loss between predictions and true values.
4. Determine the model parameters that produce the minimum loss.

Logistic Regression – The Model

- Recall our linear model for regression

$$\hat{y}_i = \sum_{j}^{p} x_{ij} \cdot \beta_j$$

- We can't use this for classification because we must predict 0 or 1.
- So, let's squash this between 0 and 1.

Logistic Regression – The Model

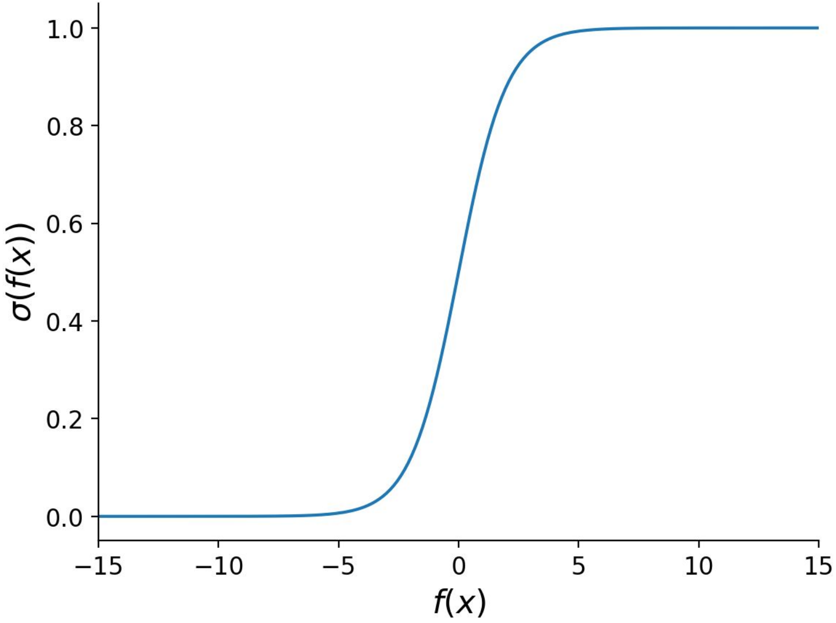
- Squash the linear model between 0 and 1.

$$\hat{y}_i = \sum_{j}^{p} x_{ij} \cdot \beta_j$$

$$f(x_i) = \sum_{j}^{p} x_{ij} \cdot \beta_j$$

$$\hat{y}_i = \sigma(f(x_i))$$

$$\sigma(f(x_i)) = \frac{1}{1 + e^{-f(x_i)}}$$

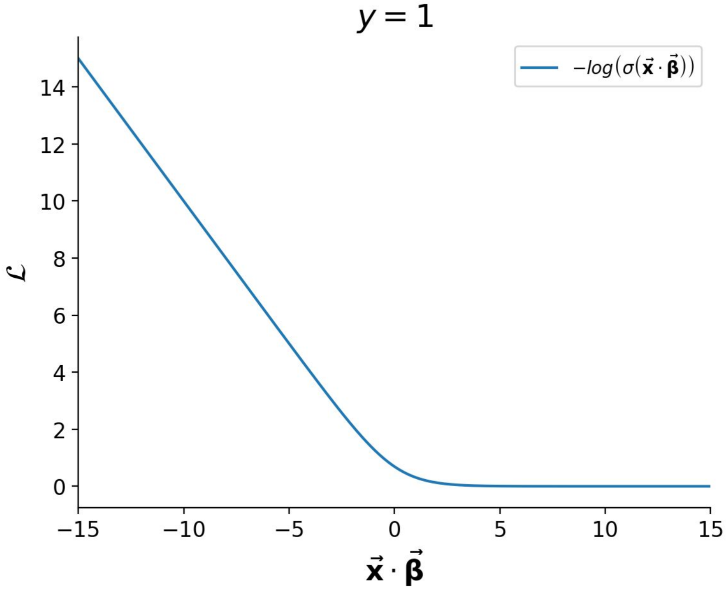


The ML Recipe

1. Think up some model
2. Feed **data** into the model and make predictions.
3. Calculate the loss between predictions and true values.
4. Determine the model parameters that produce the minimum loss.

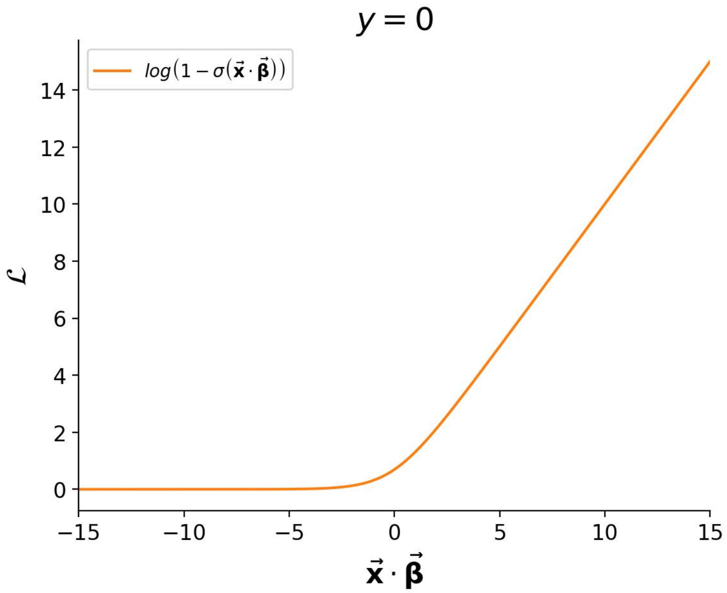
Logistic Regression – The Loss

$$\mathcal{L}\left(\vec{\beta}\right) = -\sum_{i=1}^{N} y_i \, log\left(\sigma\left(\overrightarrow{\mathbf{x_i}} \cdot \vec{\beta}\right)\right)$$



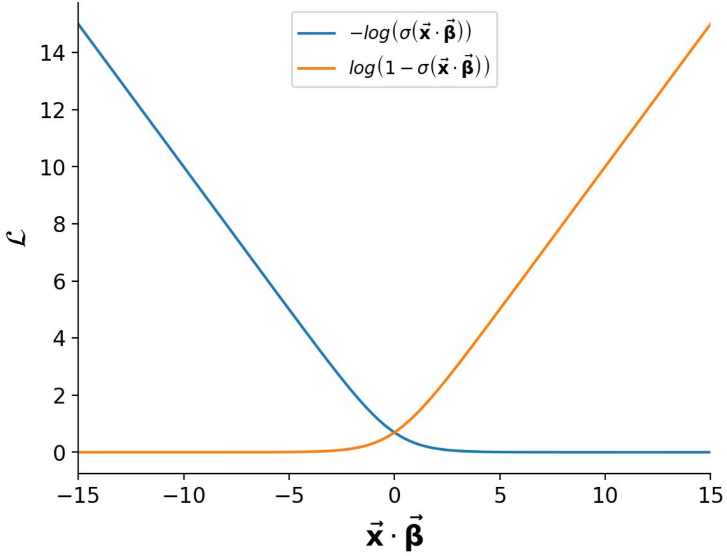
Logistic Regression – The Loss

$$\mathcal{L}\left(\vec{\beta}\right) = -\sum_{i=1}^{N} \qquad + (1 - y_i)\, log\left(1 - \sigma\left(\overrightarrow{\mathbf{x_i}} \cdot \vec{\beta}\right)\right)$$



Logistic Regression – The Loss

$$\mathcal{L}\left(\vec{\beta}\right) = -\sum_{i=1}^{N} y_i \, log\left(\sigma\left(\overrightarrow{\mathbf{x_i}} \cdot \vec{\beta}\right)\right) + (1 - y_i)\, log\left(1 - \sigma\left(\overrightarrow{\mathbf{x_i}} \cdot \vec{\beta}\right)\right)$$



The ML Recipe

1. Think up some model
2. Feed **data** into the model and make predictions.
3. Calculate the loss between predictions and true values.
4. Determine the model parameters that produce the minimum loss.

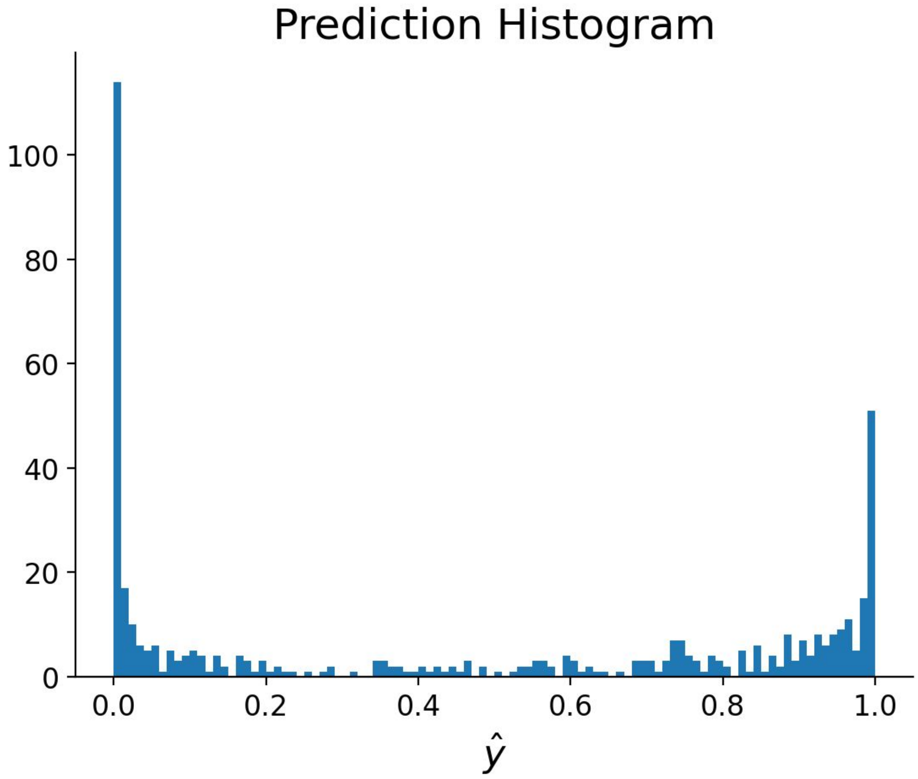
The ML Recipe

1. Think up some model
2. Feed **data** into the model and make predictions.
3. Calculate the loss between predictions and true values.
4. Determine the model parameters that produce the minimum loss.

$$\frac{\partial \mathcal{L}}{\partial \vec{\beta}} = \sum_{i=1}^{N} \left(y_i - \frac{1}{1 + e^{\left(\vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\beta}\right)}} \right) \vec{\mathbf{x}}_{\mathbf{i}}$$

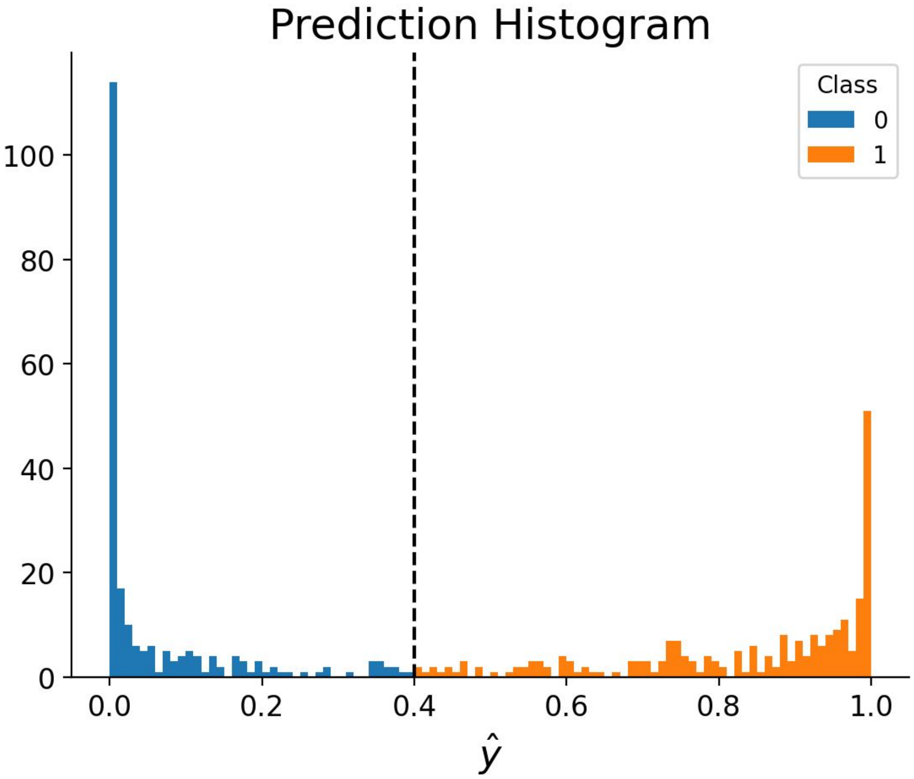
Logistic Regression – The Predictions

- Predictions lie between 0 and 1.
- They can be interpreted as probabilities.



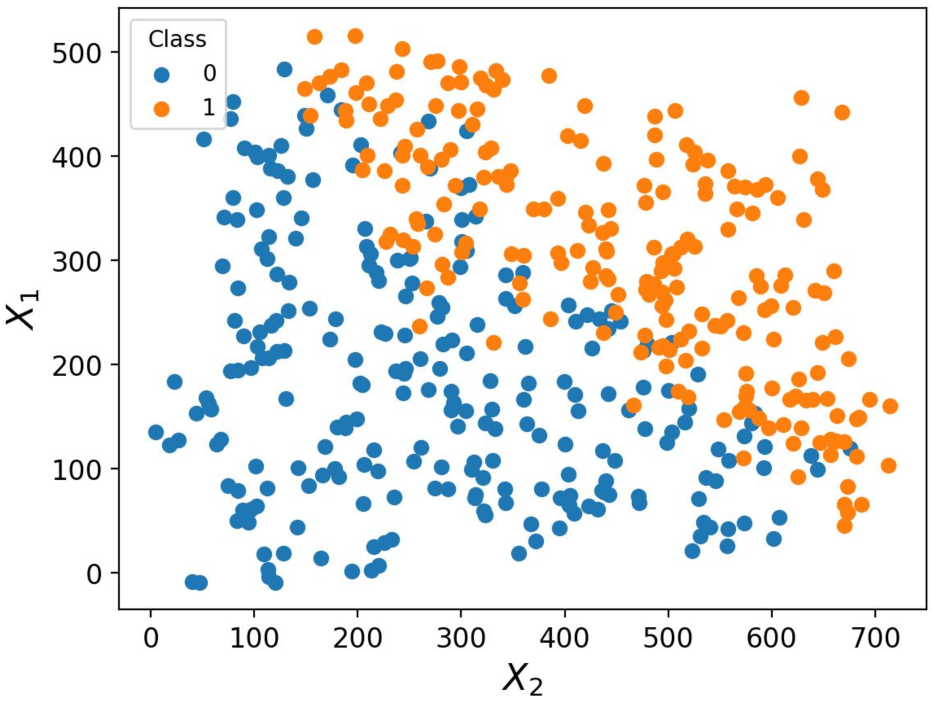
Logistic Regression – The Predictions

- Predictions lie between 0 and 1.
- They can be interpreted as probabilities.
- We pick a **threshold** to classify probabilities into classes.



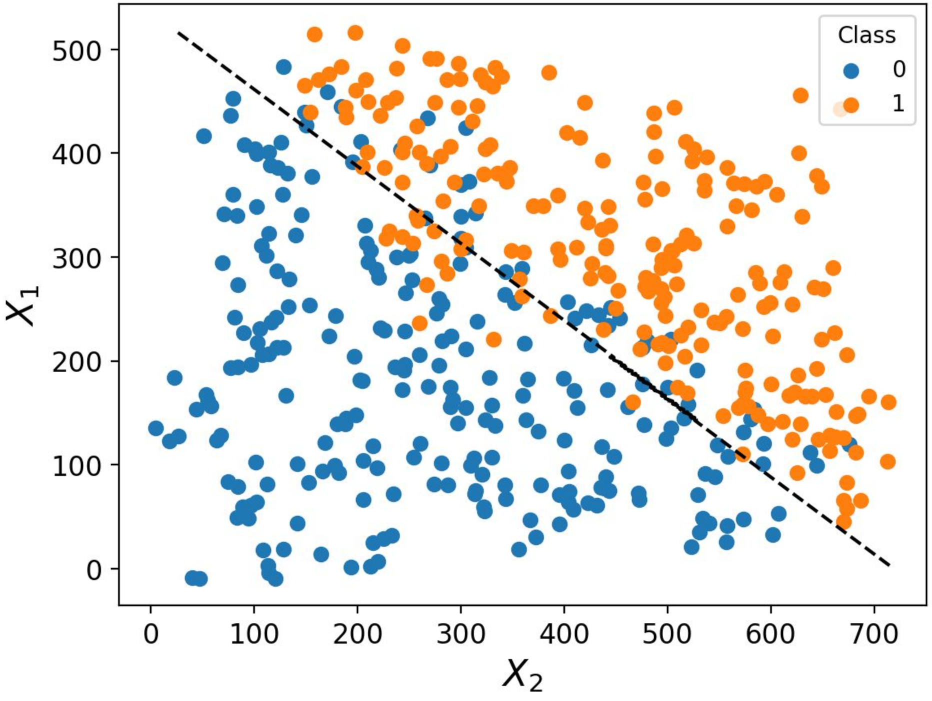
Logistic Regression – The Decision Boundary

- Imagine a classification problem with 2 features.
- For any value of X1 and X2, our model predicts a probability between 0 and 1.
- Predictions above (below) the threshold are predicted as 1 (0).



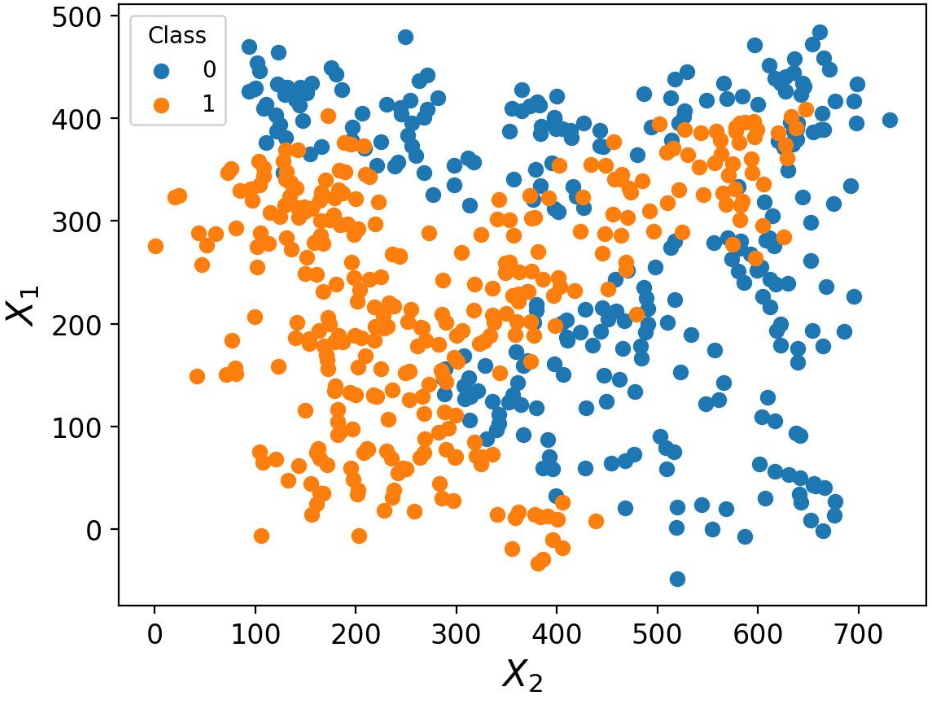
Logistic Regression – The Decision Boundary

- Imagine a classification problem with 2 features.
- For any value of X1 and X2, our model predicts a probability between 0 and 1.
- Predictions above (below) the threshold are predicted as 1 (0).
- The line that runs along the region of predictions where the prediction == threshold is the **decision boundary.**



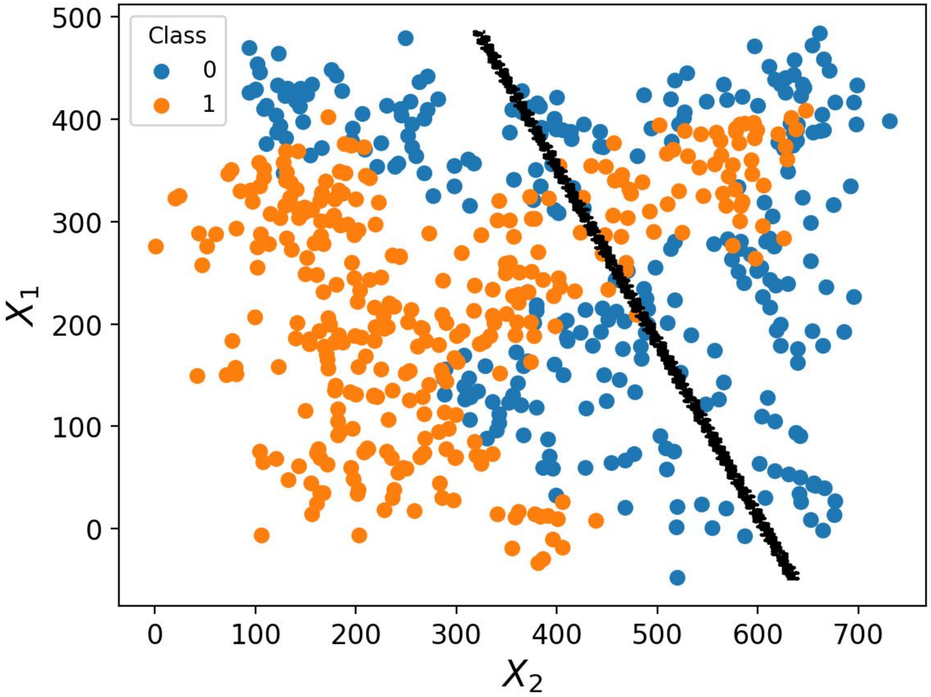
Logistic Regression – The Decision Boundary

- Problems are often nonlinear.



Logistic Regression – The Decision Boundary

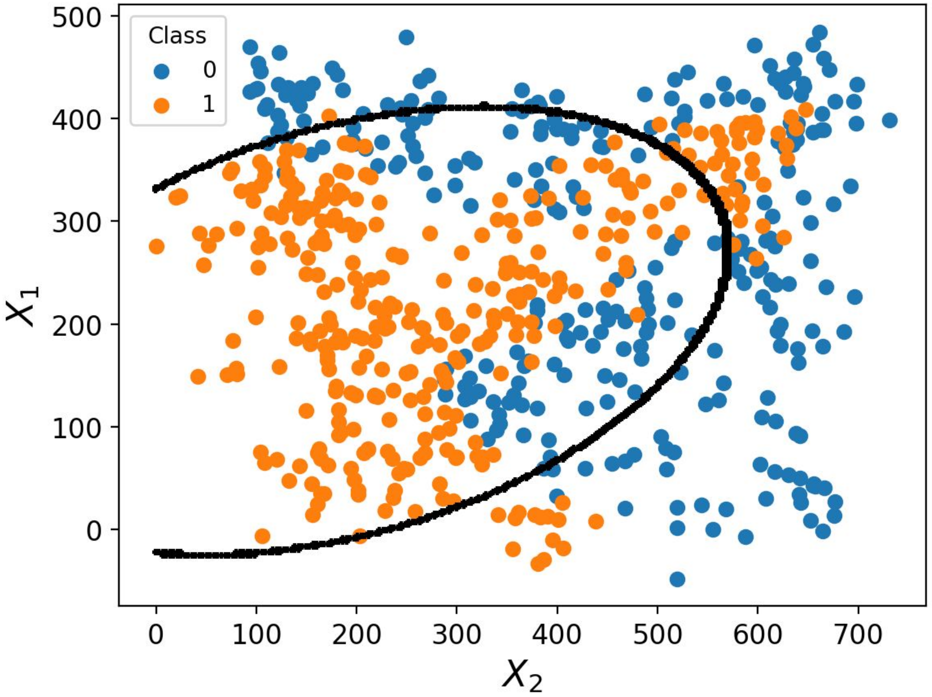
- Problems are often nonlinear.
- And linear decision boundaries don’t work well on them!



Logistic Regression – The Decision Boundary

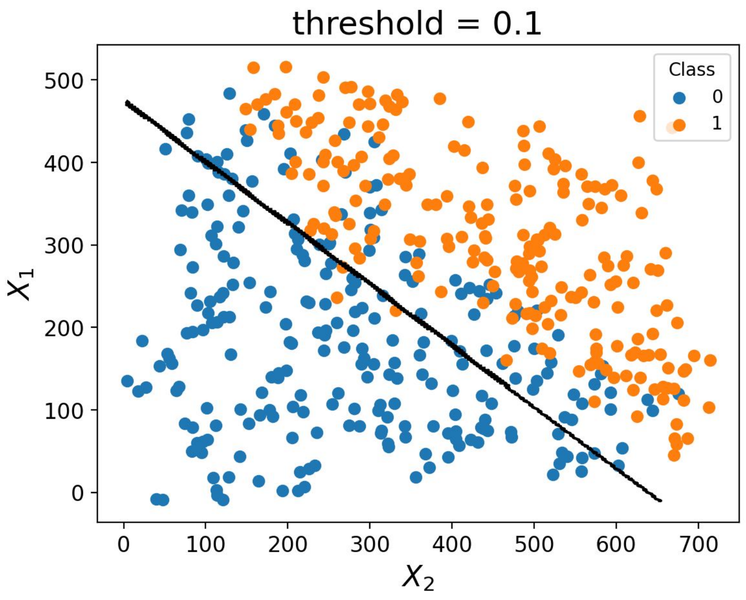
- Problems are often nonlinear.
- And linear decision boundaries don't work well on them!
- Can try creating nonlinear features:

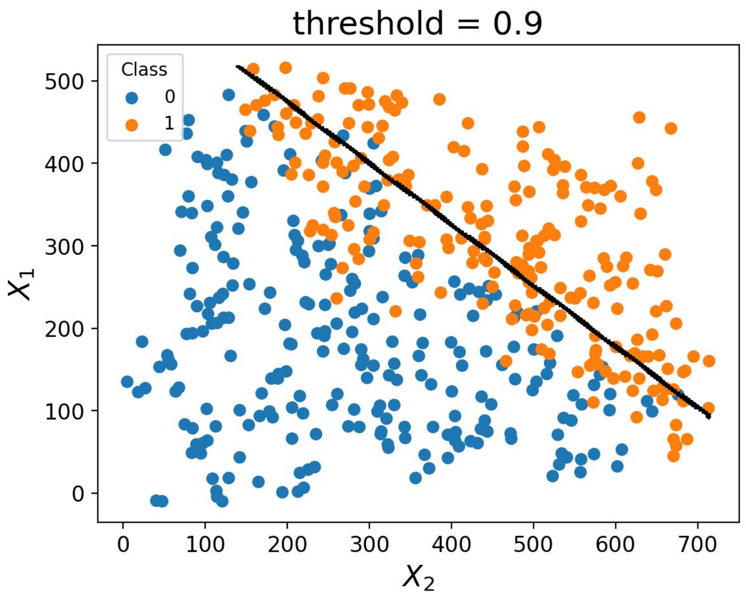
$$\hat{y}_i = \beta_0 + \beta_1 \cdot X_{i1} + \beta_2 \cdot X_{i2} + \beta_3 \cdot X_{i1} \cdot X_{i2} + \beta_4 \cdot X_{i1}^2 + \beta_5 \cdot X_{i2}^2$$



Logistic Regression – The Decision Boundary

Your threshold / decision boundary is a choice!



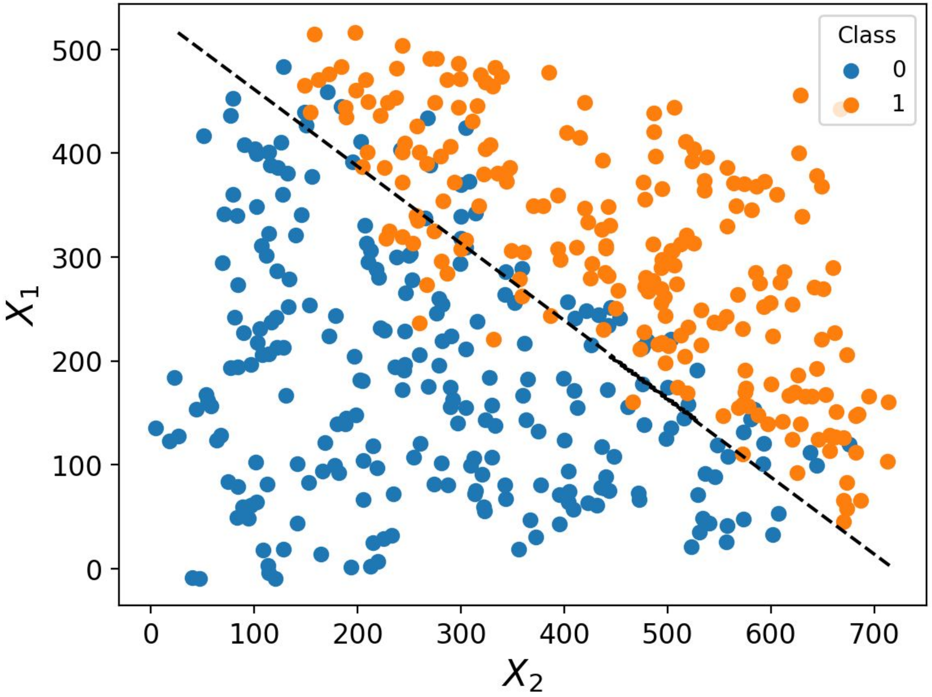


Classification Evaluation

Classification Evaluation

Confusion Matrix

Predicted Labels / True Labels	False	True
False	204	41
True	16	198

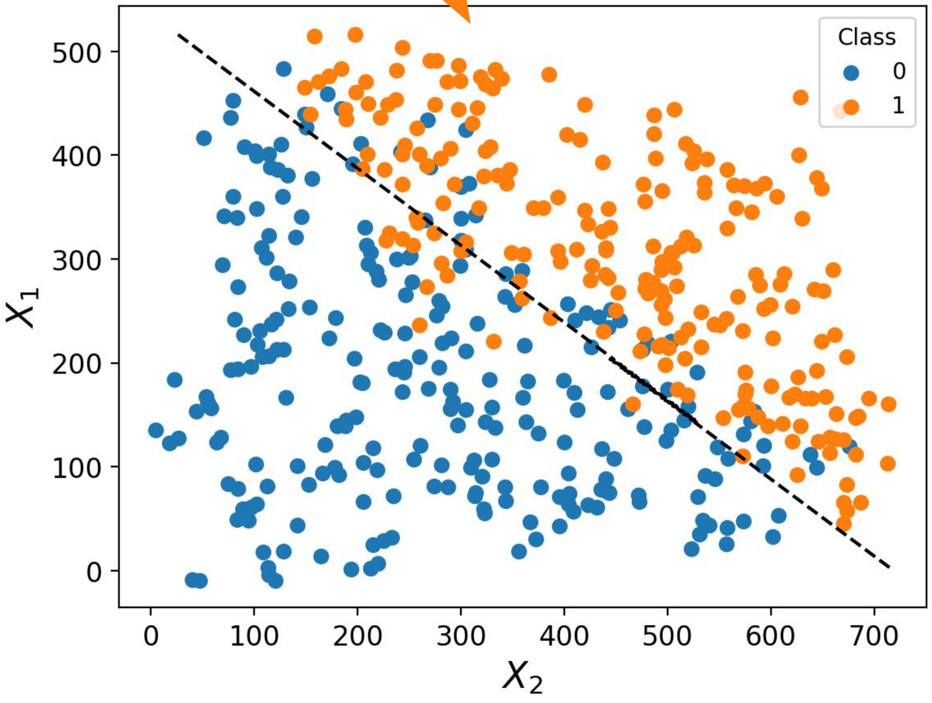


Classification Evaluation

True Positives
All orange dots on this side

Confusion Matrix

Predicted Labels / True Labels	False	True
False	204	41
True	16	198

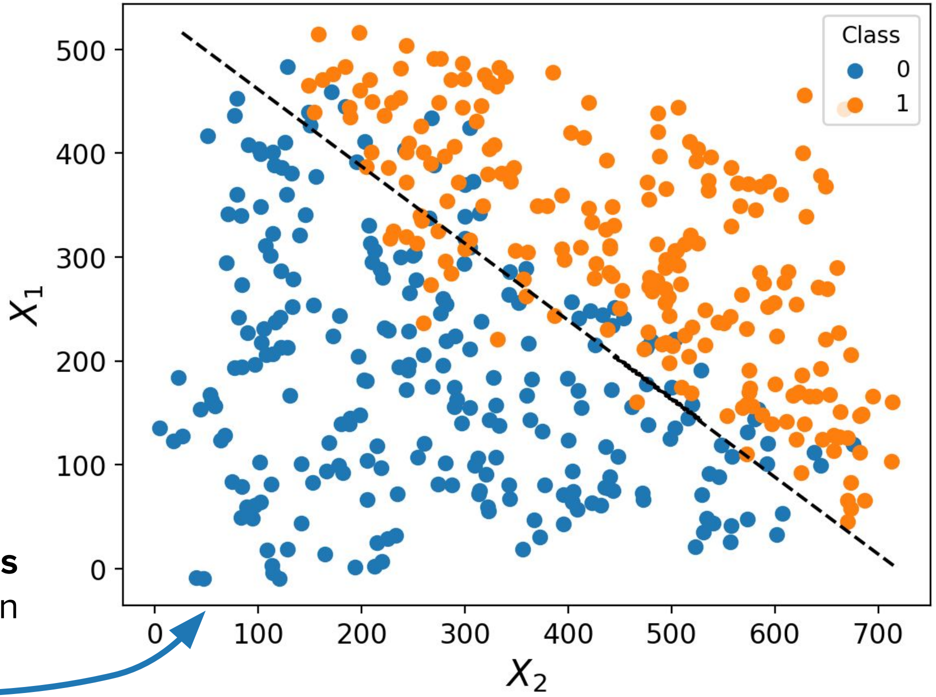


Classification Evaluation

Confusion Matrix

True Labels \ Predicted Labels	False	True
False	204	41
True	16	198

True Negatives
All blue dots on this side

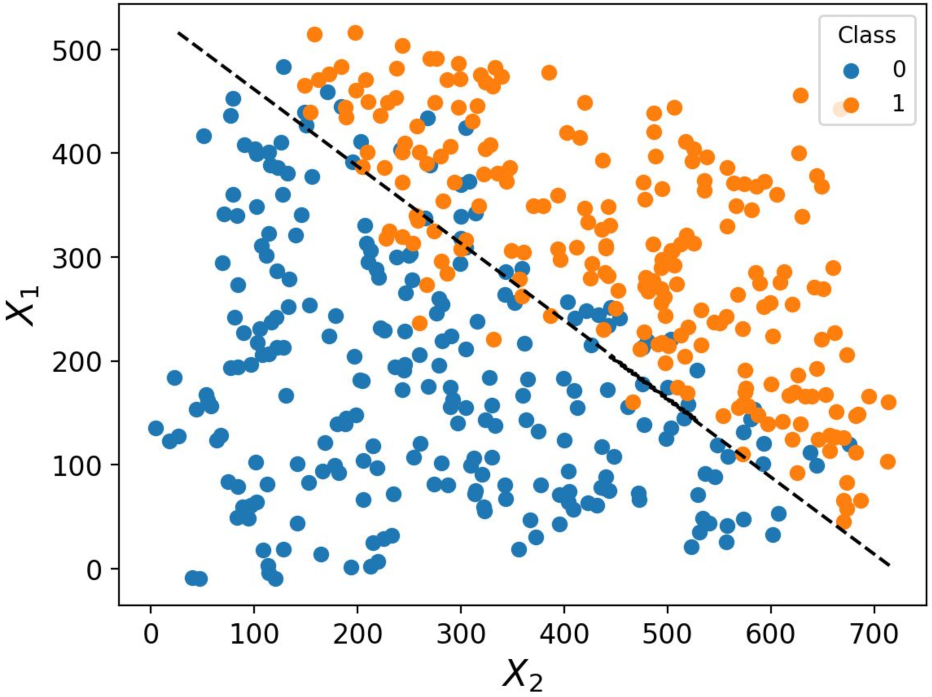


Classification Evaluation

False Positives
All blue dots on
this side

Confusion Matrix

Predicted Labels / True Labels	False	True
False	204	41
True	16	198

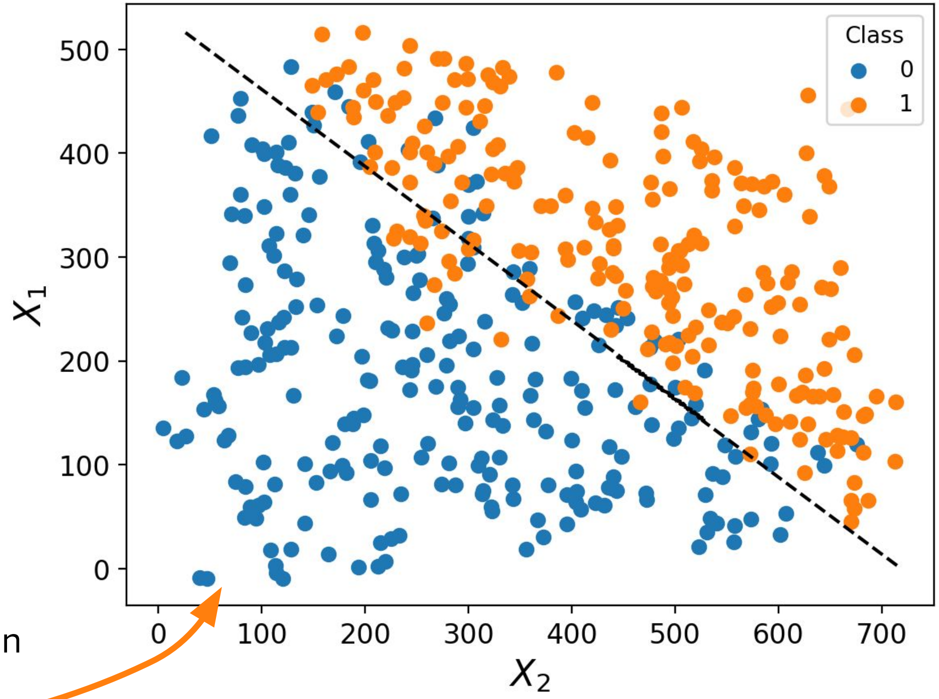


Classification Evaluation

Confusion Matrix

Predicted Labels / True Labels	False	True
False	204	41
True	16	198

False Negatives
All orange dots on this side



Classification Evaluation

Confusion Matrix

True Labels \ Predicted Labels	False	True
False	204	41
True	16	198

$$accuracy = \frac{TP + TN}{N}$$

Total samples → N

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Classification Evaluation

Confusion Matrix

True Labels \ Predicted Labels	False	True
False	204	41
True	16	198

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Precision: For all of the samples my model predicted positive, what % of them were actually positive?

Recall: Out of all positive examples in my data, what % did my model predict positive?

Classification Evaluation

Confusion Matrix

True Labels \ Predicted Labels	False	True
False	204	41
True	16	198

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

F1 Score: Harmonic mean of precision and recall

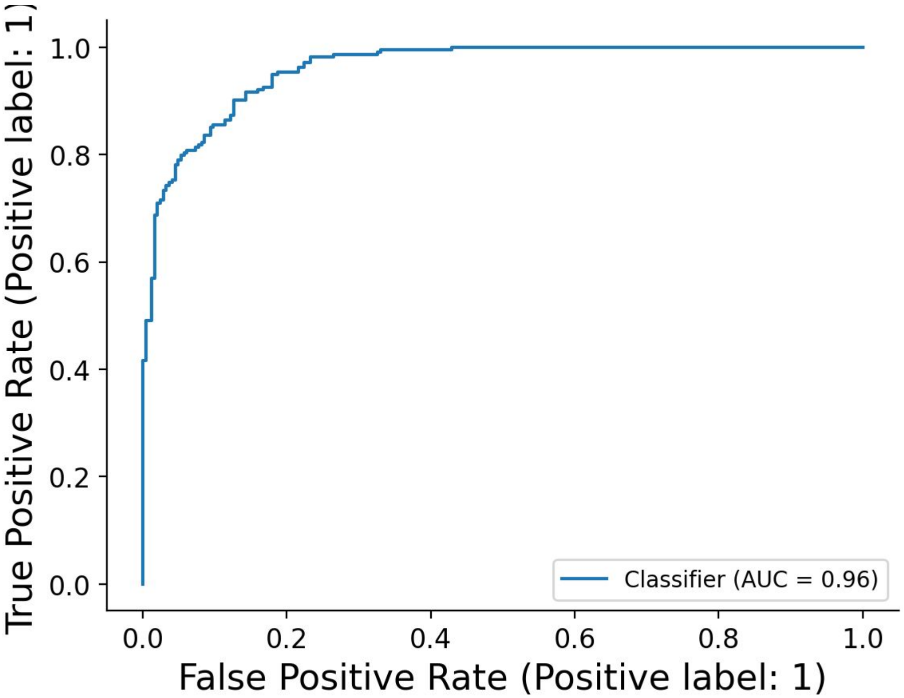
$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2\frac{precision \cdot recall}{precision + recall}$$

The ML Recipe

1. Think up some model
2. Feed **data** into the model and make predictions.
3. Calculate the loss between predictions and true values.
4. Determine the model parameters that produce the minimum loss.
5. Pick a threshold if it's a classification model.
    a. The threshold is a choice!
    b. This choice is part of your model.
    c. The confusion matrix is based on a single threshold value.

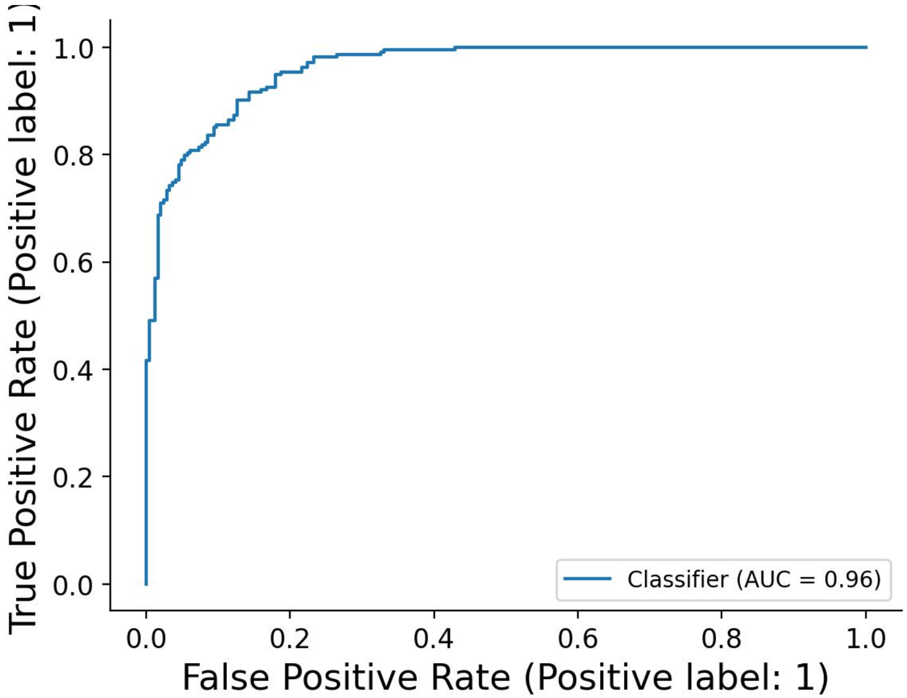
Classification Evaluation - Non-thresholded Measures

- Area Under the Receiver Operating Characteristic (ROC) Curve
  - (Sometimes just called AUC)
- TPR = Recall = TP / (TP + FN)
- FPR = TP / (FP + TN)
- Up and to the left is good
- Each point corresponds to a different threshold
- Typically, roughly concave.



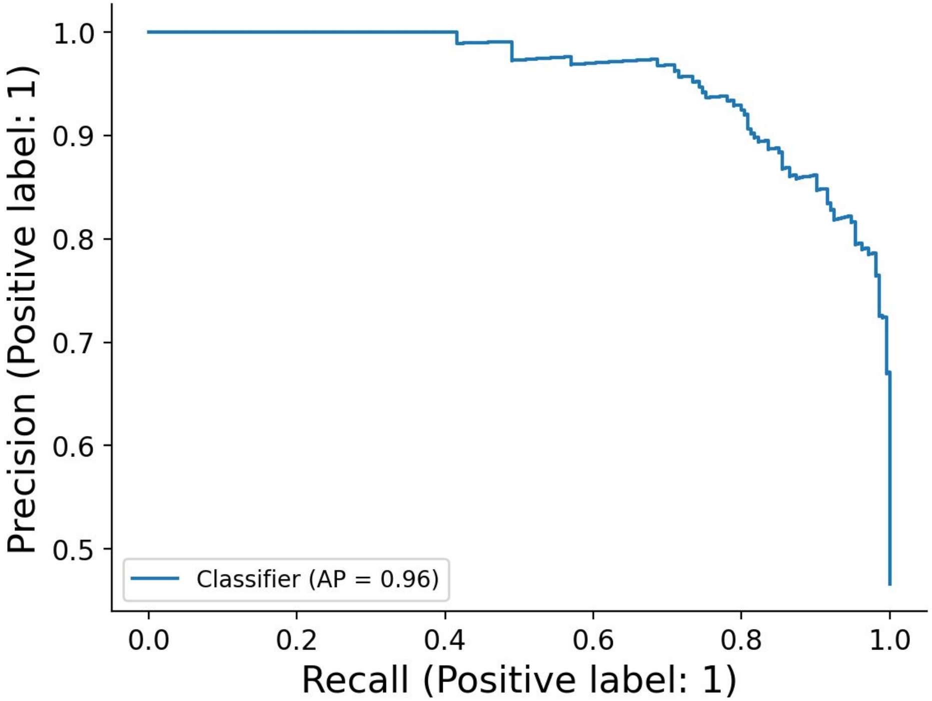
Classification Evaluation - Non-thresholded Measures

- Baseline AUROC is 0.5.
  - i.e. a random classifier
- Good, overall measure of model performance.
- Mitigates much of the impact from imbalance classes.
- Can interpret as “If you randomly draw 1 pos and 1 neg sample, what’s the probability you rank them correctly?”



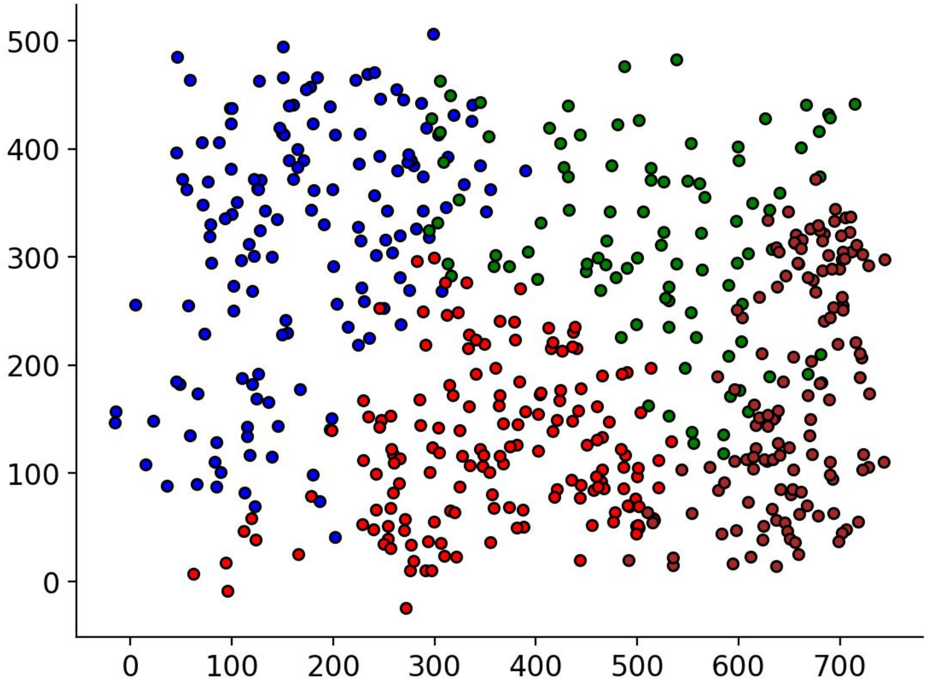
Classification Evaluation - Non-thresholded Measures

- Area Under the Precision Recall Curve (PR curve)
  - “Average Precision”
- Up and to the right is good.
- Poor models exhibit very non-concave PR curves
- Helpful for picking a threshold.

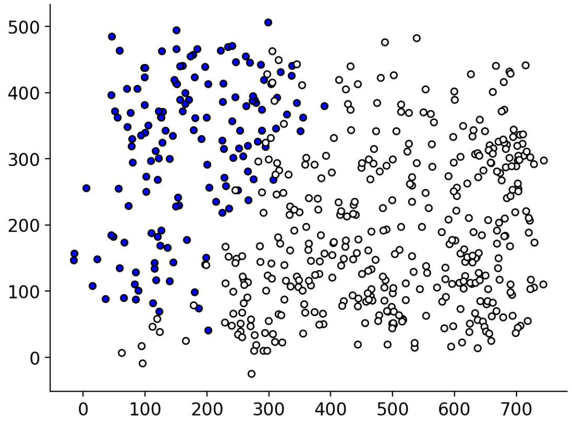


Multiclass Classification

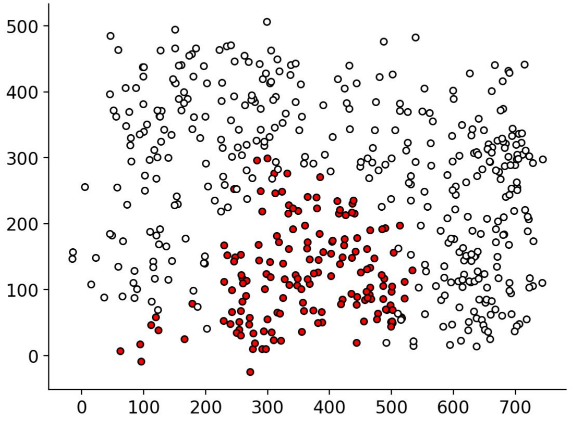
Multiclass Classification

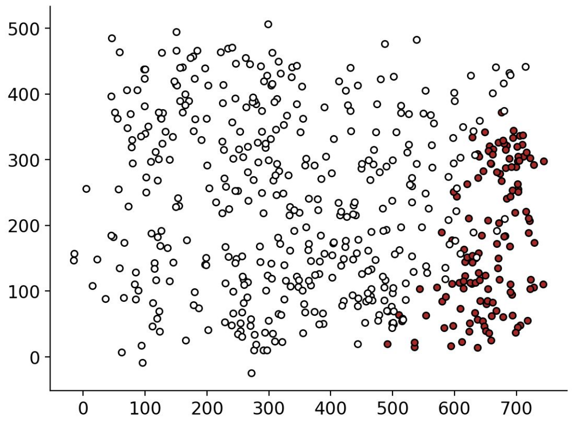


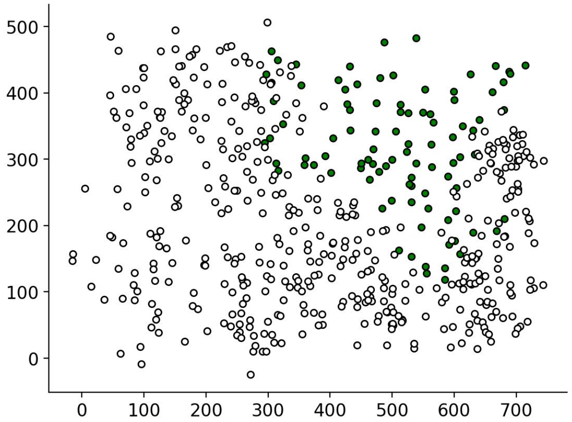
Multiclass Classification - One vs. All



- For N classes, train N binary classifiers.
- Treat each a Nth class as pos label, all other classes as negative label.

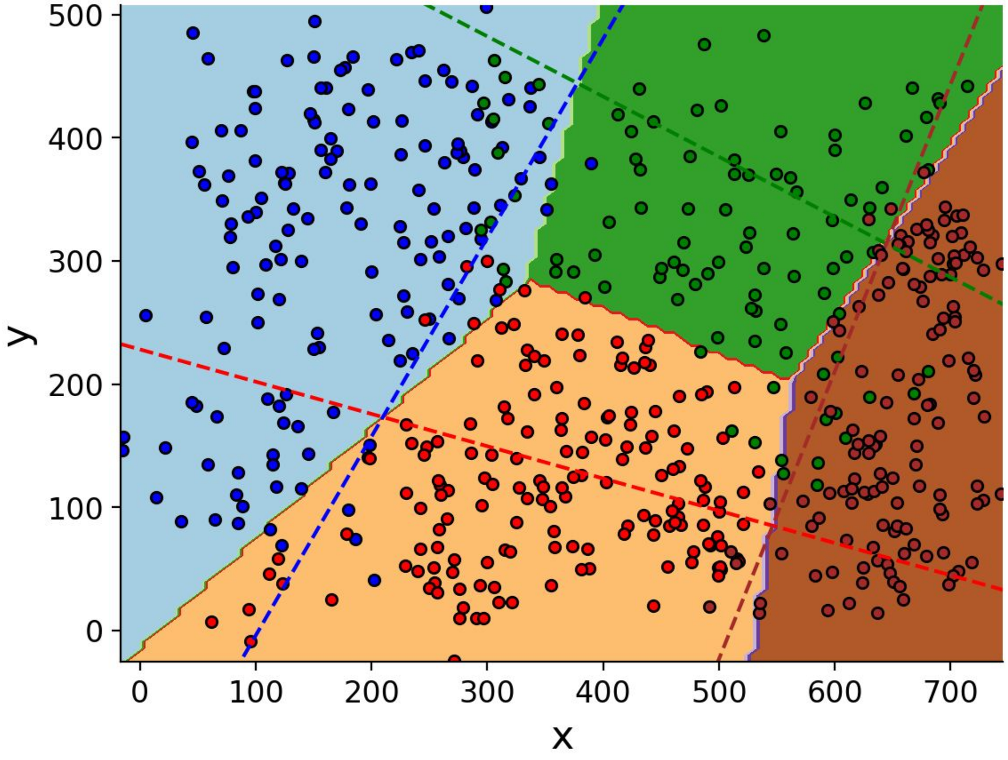






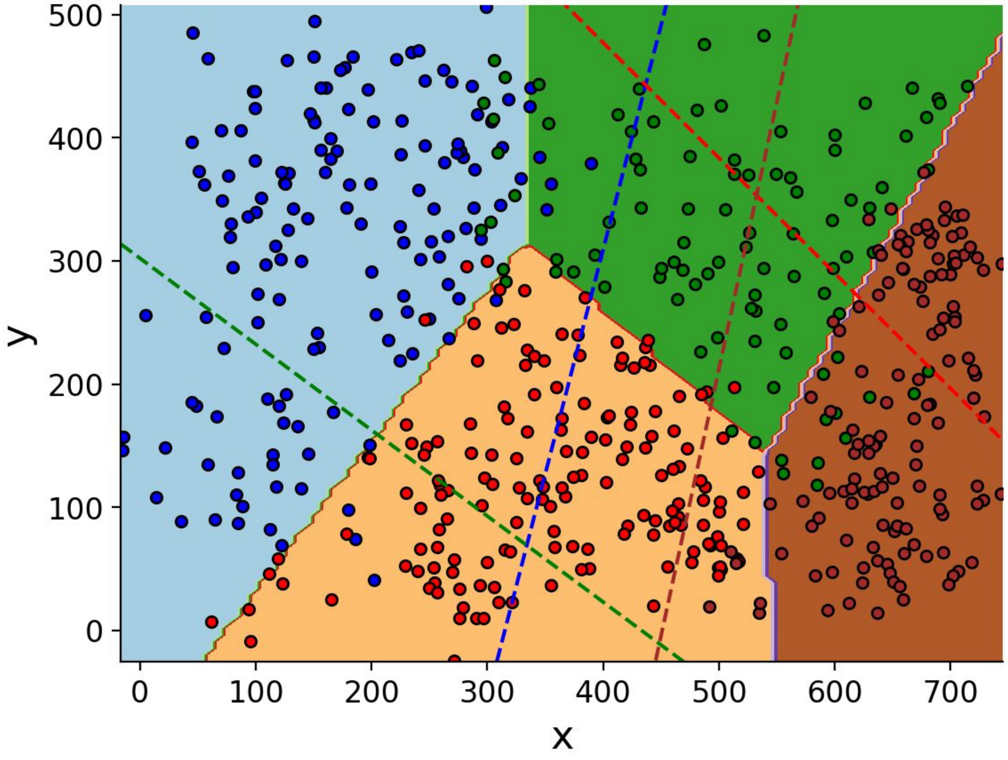
Multiclass Classification - One vs. Rest

- Max classifier score gives the predicted label.



Multiclass Classification - Multinomial

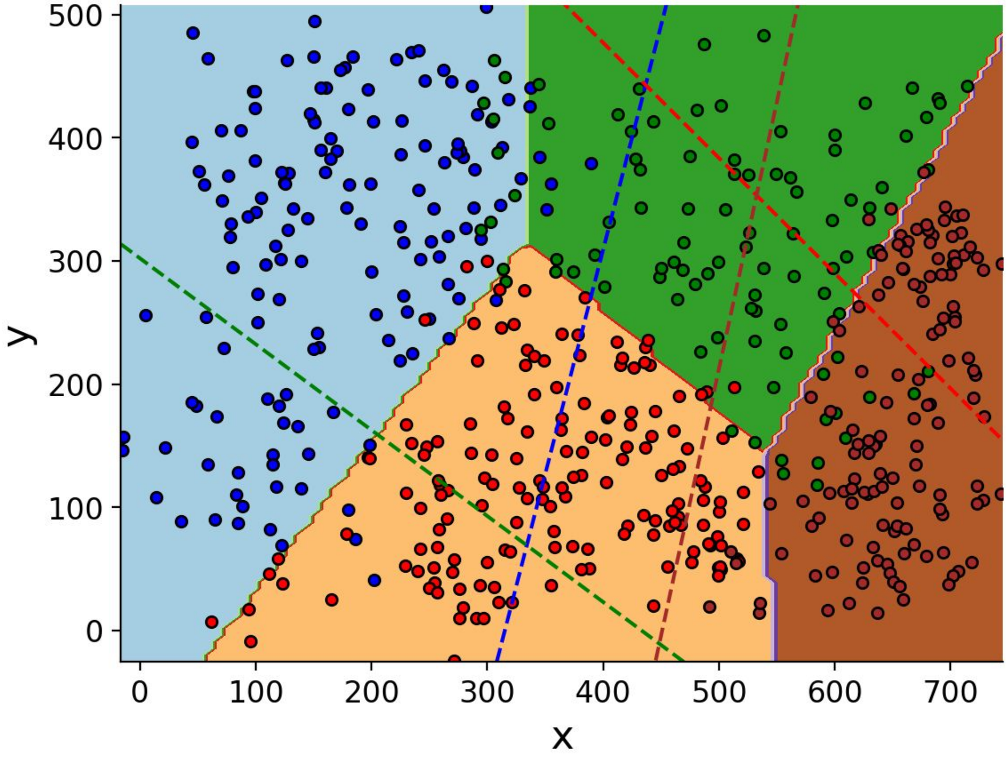
- Also called softmax or cross entropy



Multiclass Classification - Multinomial / Cross Entropy

$$\mathcal{L}_{C\mathcal{E}} = \sum_{i=1}^{N} \left(- \sum_{c=1}^{C} y_{ic} \, log \left(s \left(\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_\mathbf{c}} \right) \right) \right)$$

$$s \left(\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_{\mathbf{c}'}} \right) = \frac{e^{\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_{\mathbf{c}'}}}}{\sum_{c=1}^{C} e^{\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_\mathbf{c}}}}$$



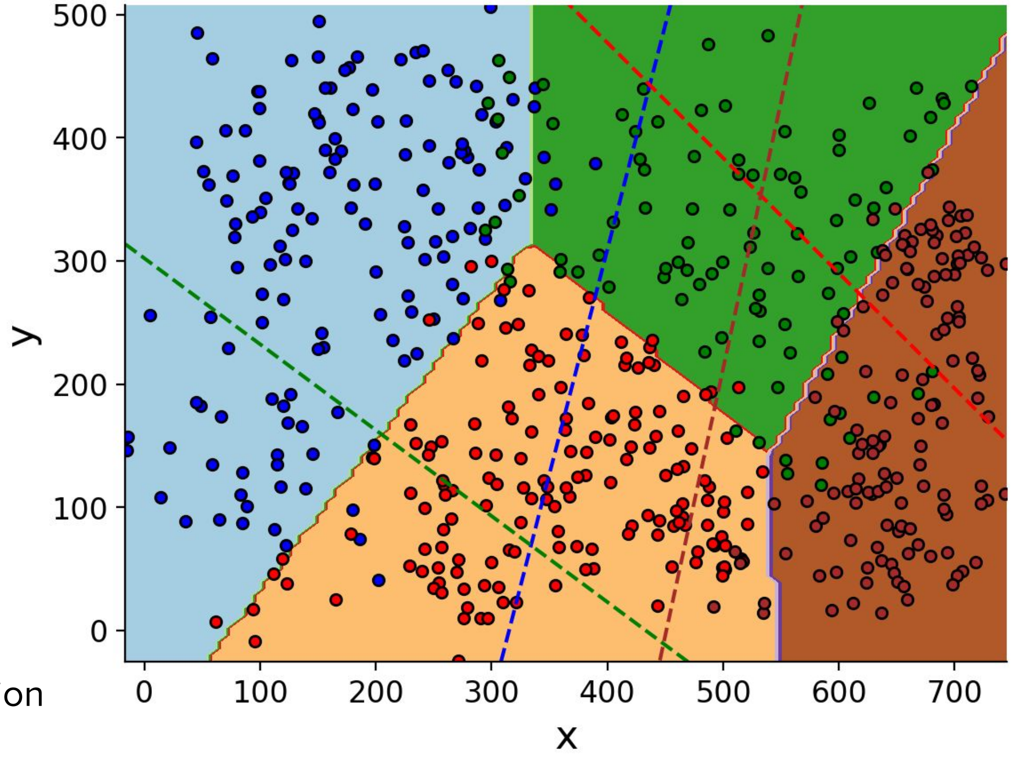
Multiclass Classification - Multinomial / Cross Entropy

$$\mathcal{L}_{C\mathcal{E}} = \sum_{i=1}^{N} \left(- \sum_{c=1}^{C} y_{ic} \, log \left(s \left(\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_{\mathbf{c}}} \right) \right) \right)$$

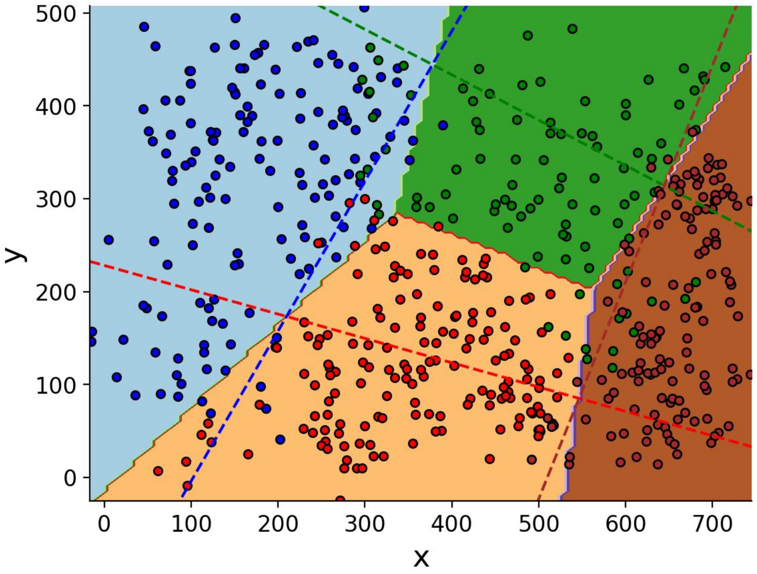
$$s \left(\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_{\mathbf{c'}}} \right) = \frac{e^{\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_{\mathbf{c'}}}}}{\sum_{c=1}^{C} e^{\overrightarrow{\mathbf{x_i}} \cdot \overrightarrow{\beta_{\mathbf{c}}}}}$$

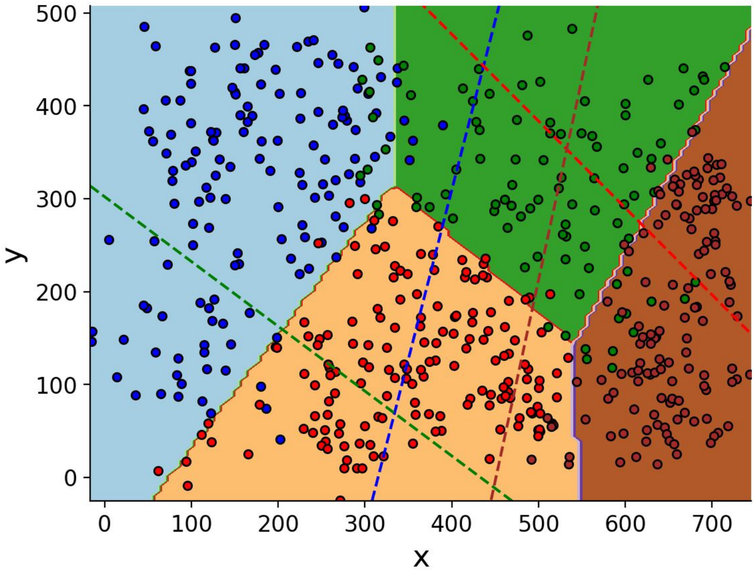
~ Probability sample belongs to class c'

Softmax function



Multiclass Classification - OVR vs Multinomial

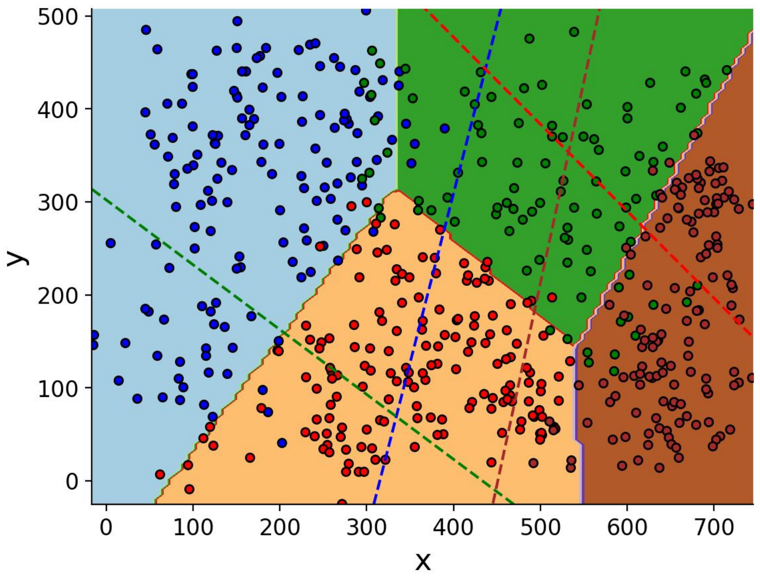




Multiclass Classification Evaluation

Confusion Matrix

Predicted Labels / True Labels	0	1	2	4
0	135	6	6	0
1	9	71	3	13
2	6	5	150	0
4	0	5	5	121



Multiclass Classification Evaluation

Accuracy is now harder!

Binary: For 50/50 labels, random classifier gets 50% accuracy

Multiclass: for equal numbers of C labels, random classifier gets 1 / C accuracy

For 4 classes, random classifier gets 25% accuracy.

Multiclass Classification Evaluation

- Precision, recall, and F1 all have multiclass equivalents.
- You can compute them for any individual class.

$$Precision_c = \frac{TP_c}{TP_c + FP_c}$$

- For computing across all classes:
  - Macro-average: Calculate metric for each class and then take the average across all classes.

$$\cancel{Precision = \frac{\sum_c TP_c}{\sum_c TP_c + FP_c}}$$

  - Micro-average: Compute numerators and denominators for all classes.

$$Precision = \frac{\sum_c TP_c}{\sum_c TP_c + FP_c}$$

Language Models are Multiclass Classifiers

- Starting Data: “Sounds like fun!”

Language Models are Multiclass Classifiers

- Starting Data: “Sounds like fun!”
- Break it up into data + answers where we try to predict the next word:

Data	Answer
Sounds	like
Sounds like	fun
Sounds like fun	!

Language Models are Multiclass Classifiers

- Starting Data: “Sounds like fun!”
- Break it up into data + answers where we try to predict the next word:

Data	Answer
Sounds	like
Sounds like	fun
Sounds like fun	!

- One-hot-encode words into numbers:

0	0	1	0	1	1	0	1	…
hi	you	like	dog	fun	sounds	me	!	

Language Models are Multiclass Classifiers

- Starting Data: “Sounds like fun!”
- Break it up into data + answers where we try to predict the next word:

Data	Answer
Sounds	like
Sounds like	fun
Sounds like fun	!

- One-hot-encode words into numbers:

0	0	1	0	1	1	0	1	…
hi	you	like	dog	fun	sounds	me	!	

- Feed into a model, treat all words as classes, and predict the answer word (class)