Classification I

Classification Overview

- Classification Model
  - A type of machine learning model for distinguishing among two or more discrete classes.
  - For example, a natural language processing classification model could determine whether an input sentence was in French, Spanish, or Italian.
- Regression Model
  - A type of model that outputs continuous (typically, floating-point) values.

Popular Classification Algos

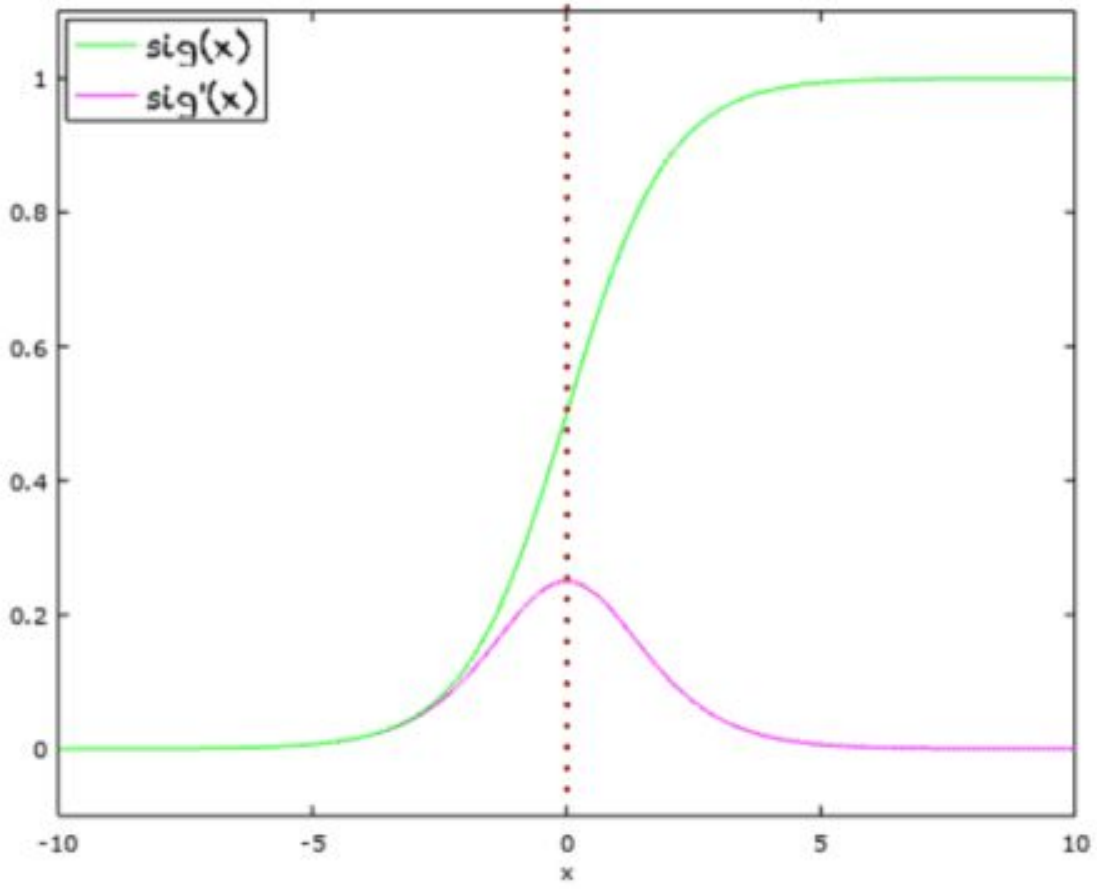
- Logistic Regression
- Naive Bayes
- K-Nearest Neighbors
- Decision Trees
- Support Vector Machines

Logistic Regression

Logistic Regression Output

- Many problems require a probability estimate as output.
- Logistic regression returns a **probability**.
- You can use the returned probability in either of the following two ways:
  - "As is"
  - Converted to a binary category (classification, e.g. "hot dog" or "not hot dog"). How?
- How do we guarantee that the output is always a probability, e.g. between 0 and 1?

Sigmoid Function



Plot of $\sigma(x)$ and its derivate $\sigma'(x)$

Domain: $(-\infty, +\infty)$

Range: $(0, +1)$

$\sigma(0) = 0.5$

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

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

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Sigmoid Function

- Properties
  - Domain: $(-\infty, +\infty)$
  - Range: (0, +1)
  - $\sigma(0) = 0.5$
  - The function is monotonically increasing.
  - The function is continuous everywhere.
  - The function is differentiable everywhere in its domain.
  - Numerically, it is enough to compute this function's value over a small range of numbers, e.g., [-10, +10]. For values less than -10, the function's value is almost zero. For values greater than 10, the function's values are almost one.
- Also known as a "squashing function", since no matter how big the input (domain), output is always between 0 and 1.
- Also used as an Activation Function in Neural Networks.

How does it work?

- Let
  - ***z*** represent the output of the linear layer of a model trained with logistic regression
  - ***y'*** is the output of the logistic regression model for a particular example.

$$z = b + w_1 x_1 + w_2 x_2 + \ldots + w_N x_N \qquad y' = \frac{1}{1 + e^{-z}}$$

  - The w values are the model's learned weights, and b is the bias.
  - The x values are the feature values for a particular example.

Logistic Regression Example

Learned weights

$$b = 1$$
$$w_1 = 2$$
$$w_2 = -1$$
$$w_3 = 5$$

Example values

$$x_1 = 0$$
$$x_2 = 10$$
$$x_3 = 2$$

z calculation

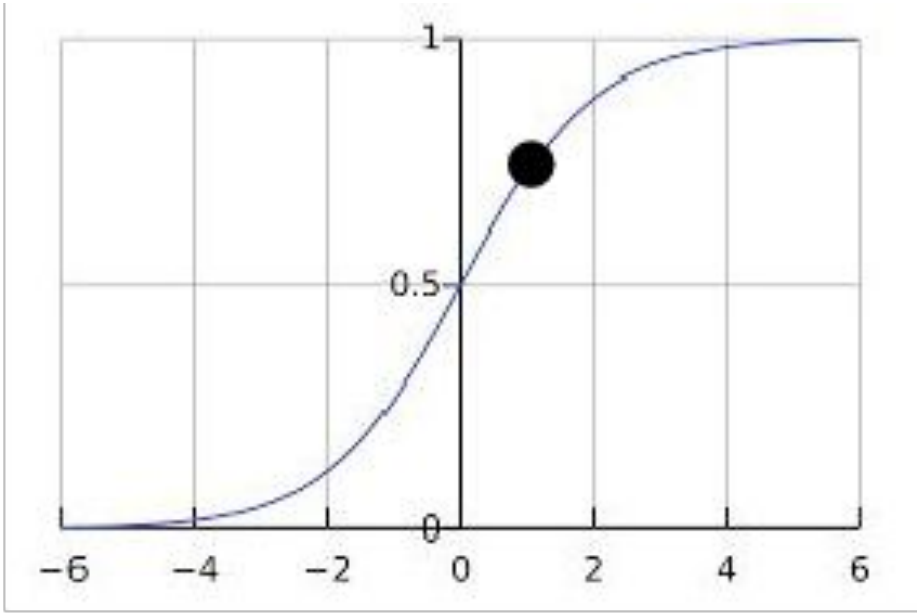
$$b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

$$(1) + (2)(0) + (-1)(10) + (5)(2) = 1$$

Prediction for this example

$$y' = \frac{1}{1 + e^{-1}} = 0.731$$

Probability Output



$$z = (b + w_1 x_1 + w_2 x_2 + \ldots w_N x_N)$$

Applications in Finance

- Credit Score
- Risk Analysis

Academy of Accounting and Financial Studies Journal (Print ISSN: 1096-3685; Online ISSN: 1528-2635)

Research Article: 2020 Vol: 24 Issue: 3

The Application of The Logistic Model in Analyzing the Risk of Bankruptcy of Small and Medium Enterprises (SMEs): A Case Study

Dao Thi Thanh Binh, Hanoi University

Bui Thi Thu Loan, Hanoi University of Industry

Vu Thi Kim Anh, Trade Union University

Abstract

Using the data of 46 unlisted small and medium enterprises divided into two categories, consisting of 23 bankrupted and non-bankrupted enterprises, the study examined some aspects of bankruptcy risk prediction based on the logistic regression model to identify the most important financial factors related to viability of small businesses. The most notable findings of the study are that businesses in suburban areas are at a higher risk of bankruptcy than firms concentrated in the central districts. The age of enterprises also affects the risk of bankruptcy. For financial indicators, the research results show that economic profitability, firm size, revenue growth and sales to capital investment are important and meaningful factors related to forecasting the exhaustion. Small businesses truly face financial crises in the context of the transition economy, developing along with the young financial market in Vietnam today. Based on the research results, some limitations and further research directions are also proposed.

Other Uses

- Although logistic regression is often used in binary classification problems, logistic regression can also be used in multi-class classification problems (where it becomes called multi-class logistic regression or multinomial regression).
- Further Reading
  - https://towardsdatascience.com/multiclass-logistic-regression-from-scratch-9cc0007da372
  - https://machinelearningmastery.com/multinomial-logistic-regression-with-python/
  - https://www.cs.princeton.edu/courses/archive/spring16/cos495/slides/ML_basics_lecture7_multiclass.pdf

Log Loss

- The loss function for logistic regression is **Log Loss**, which is defined as follows:

$$\text{Log Loss} = \sum_{(x,y)\in D} -y\log(y') - (1-y)\log(1-y')$$

where:

  - D is the data set containing many labeled examples, which are (x,y) pairs.
  - y is the label in a labeled example. Since this is logistic regression, every value of y must either be 0 or 1.
  - y' is the predicted value (somewhere between 0 and 1), given the set of features in x.

Regularization

- Extremely important in Logistic Regression
  - Without it, the asymptotic nature of logistic regression would keep driving loss towards 0 in high dimensions
- Most Logistic Regression models use one of the following to dampen model complexity
  - L_2 regularization
  - Early stopping, that is, limiting the number of training steps or the learning rate.

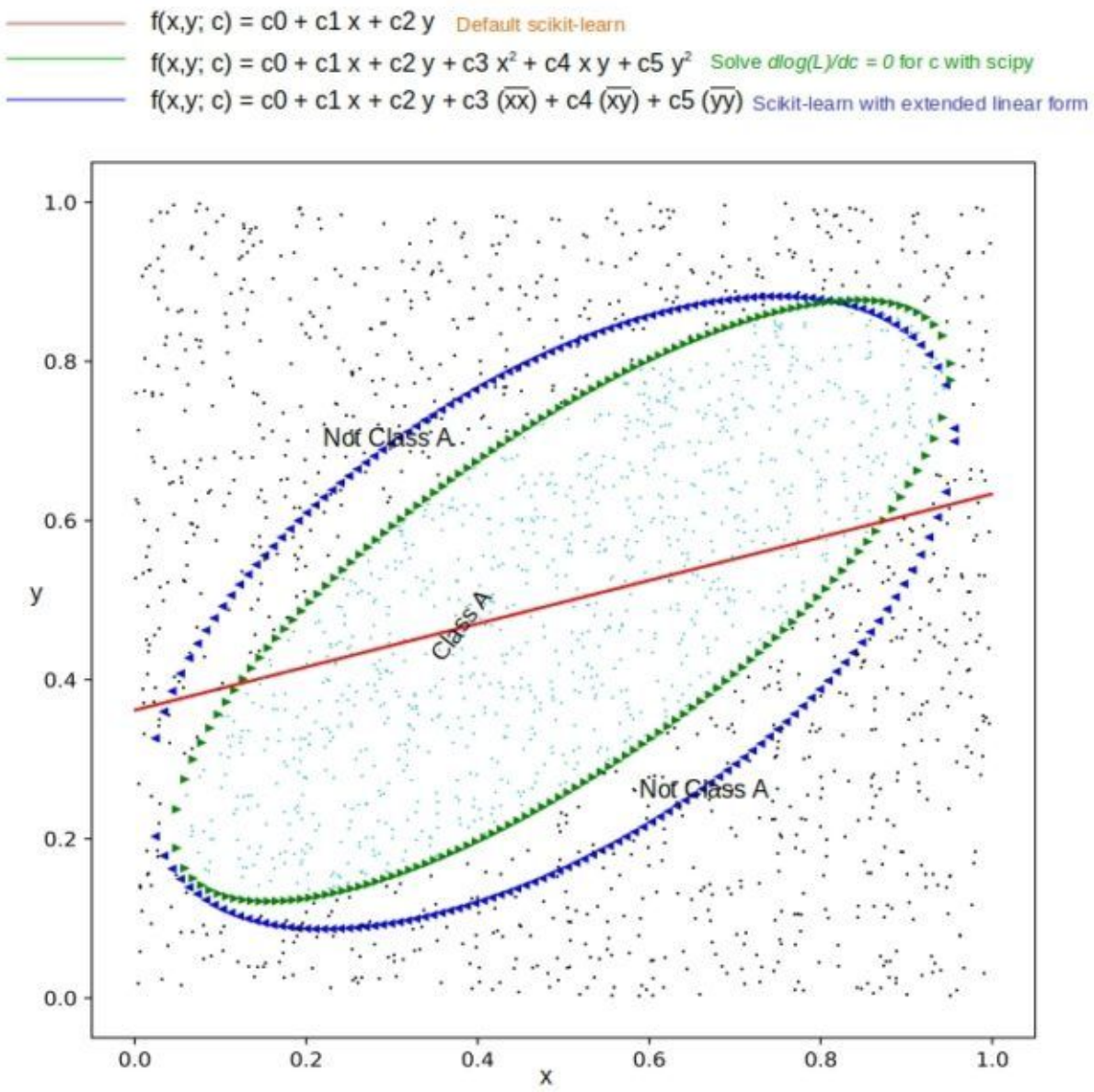
Must Logistic Regression be a linear model?

- Must z always be a linear polynomial?

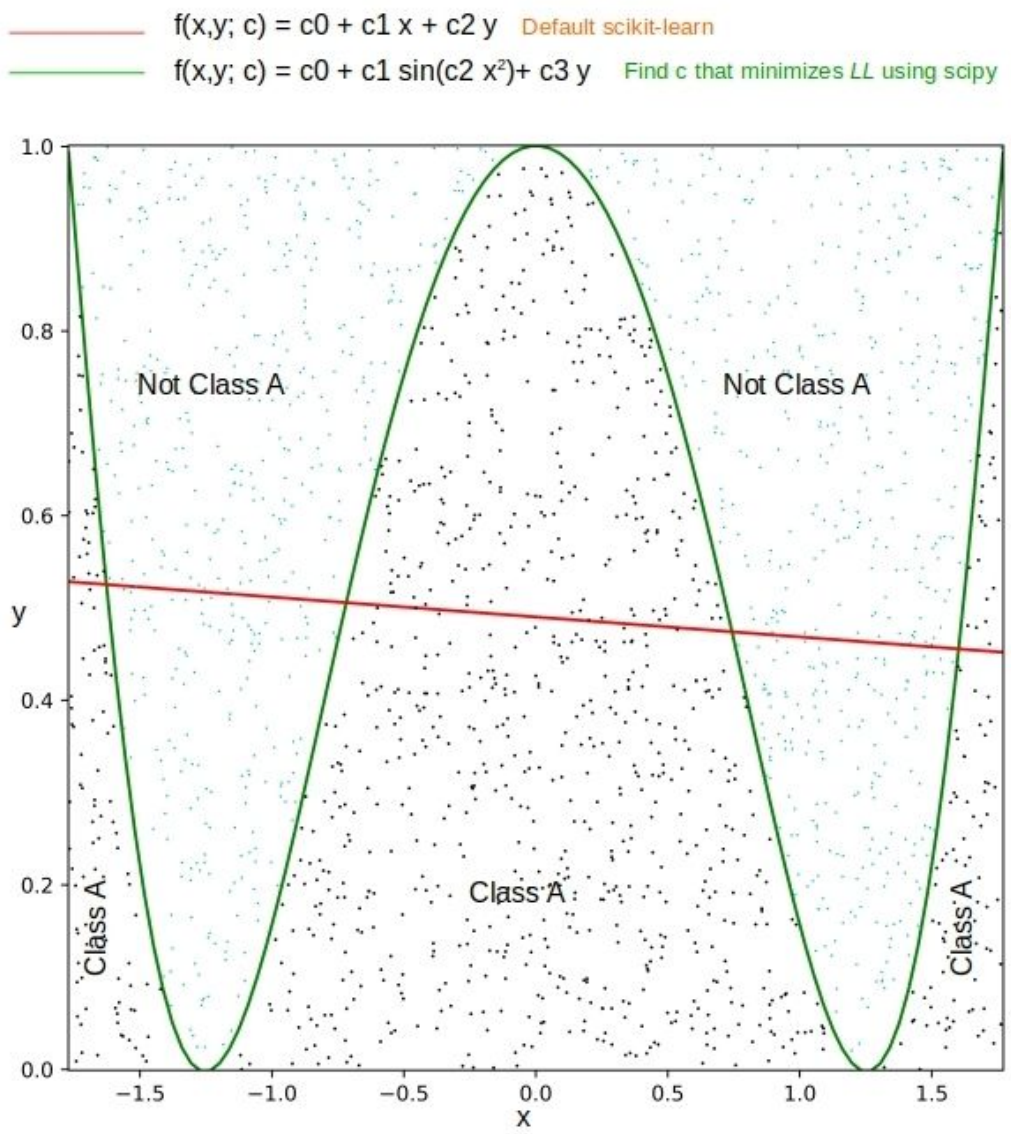
$$z = b + w_1 x_1 + w_2 x_2 + \ldots + w_N x_N \qquad y' = \frac{1}{1 + e^{-z}}$$

- No. It depends on the decision boundary.
  - Logistic regression has traditionally been used to come up with a hyperplane that separates the feature space into classes.
  - But if we suspect that the decision boundary is nonlinear we may get better results by attempting some nonlinear functional forms.
  - Solving for the model parameters can be more challenging but the optimization modules in scipy can help.
- Further reading:
  - http://xplordat.com/2019/03/13/logistic-regression-as-a-nonlinear-classifier/

Non-Linear Logistic Regression



Non-Linear Logistic Regression



THRESHOLDING

Thresholding

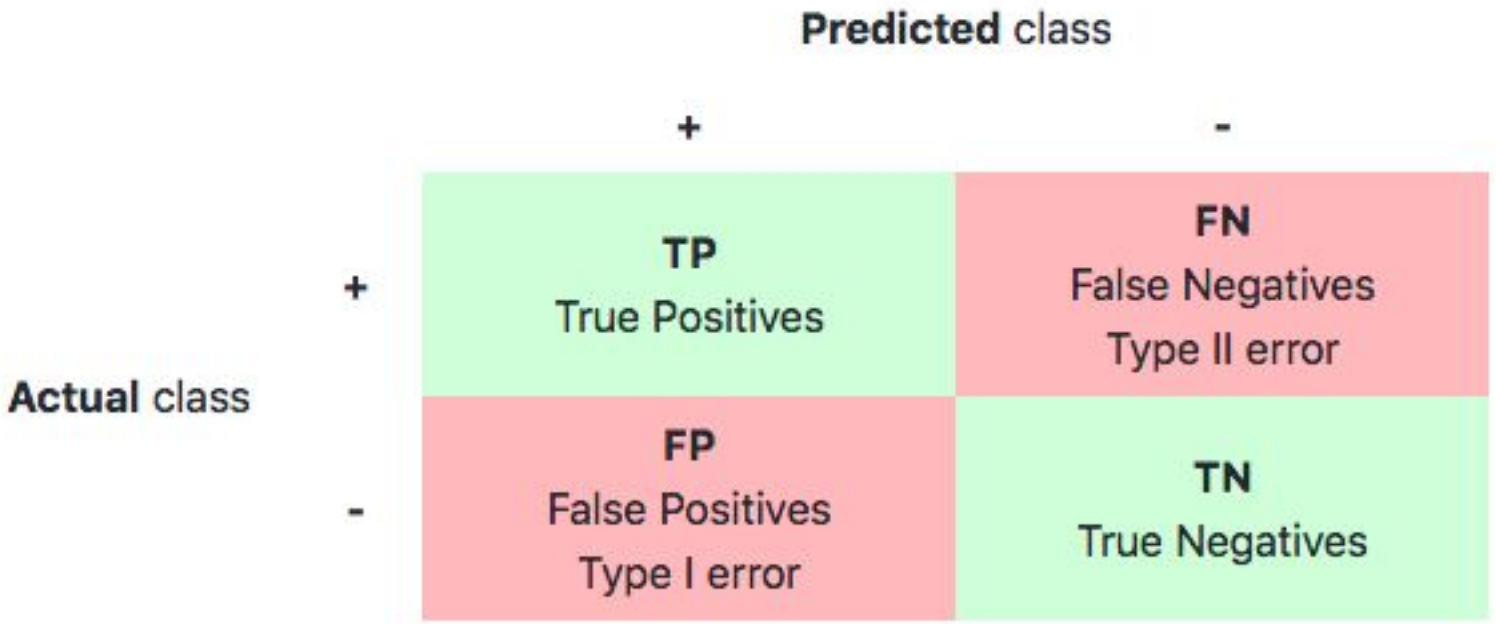
- We can use the probability output from Logistic Regression for classification
- Set a probability threshold to distinguish between the classes
    - Known as the **Classification Threshold** or Decision Threshold
  - A value above the threshold indicates one class
  - A value below the threshold indicates the other class
- The classification threshold is problem dependent, so you must tune the model to find the best one (yielding highest accuracy without overfitting)
  - Tuning the threshold is different than tuning other model parameters

Assessing Mistakes

- Part of choosing a threshold is assessing how much you'll suffer for making a mistake.
- A **true positive** is an outcome where the model *correctly* predicts the *positive* class.
- Similarly, a **true negative** is an outcome where the model *correctly* predicts the *negative* class.
- A **false positive** is an outcome where the model *incorrectly* predicts the *positive* class.
- And a **false negative** is an outcome where the model *incorrectly* predicts the *negative* class.

Confusion Matrix

- The confusion matrix is used to have a more complete picture when assessing the performance of a model.



Metrics

For binary classification:

Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual negative sample
F1 score	$\frac{2TP}{2TP + FP + FN}$	Hybrid metric useful for unbalanced classes

Precision

- Precision is the number of True Positives divided by the number of True Positives and False Positives.
- Put another way, it is the number of **positive predictions divided by the total number of positive class values predicted**.
- It is also called the Positive Predictive Value (PPV).
- Precision can be thought of as a measure of a classifiers exactness.
- A low precision can also indicate a large number of False Positives.

Recall

- Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives.
- Put another way it is the **number of positive predictions divided by the number of positive class values** in the test data.
- It is also called Sensitivity or the True Positive Rate.
- Recall can be thought of as a measure of a classifiers completeness.
- A low recall indicates many False Negatives.

Accuracy

- **Don’t be tricked by high accuracy**
- Accuracy alone doesn't tell the full story when you're working with a **class-imbalanced data set**, where there is a significant disparity between the number of positive and negative labels.
- Sometimes it may be desirable to select a model with a lower accuracy because it has a greater predictive power on the problem. This is called the Accuracy Paradox.

Class-Imbalanced Data Example

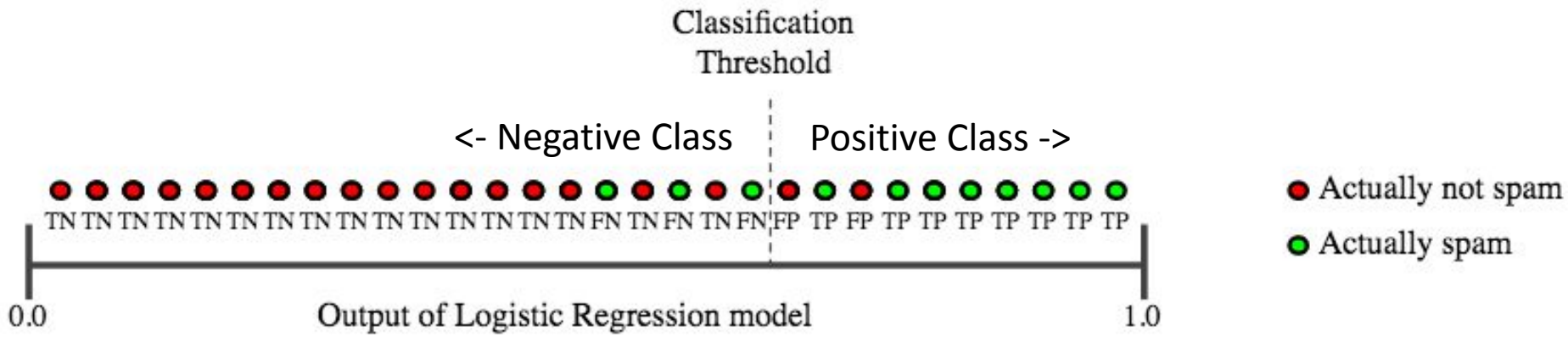
True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

- What is the fraction of predictions our model got right?
  - Accuracy = (tp + tn) /(tp+tn+fp+fn) = (1+90) / (1+90+1+8) = 0.91
- What proportion of positive identifications was actually correct?
  - Precision = tp/(tp + fp) = (1 / (1+1) = 0.5
- What proportion of actual positives was identified correctly?
  - Recall = tp / (tp + fn) = 1 / (1+8) = 0.11
    - For example, if this was a model for tumor detection, it correctly identifies only 11% of all malignant tumors.
  - A model that produces no false negatives has a recall of 1.0.

Precision and Recall: A Tug of War

- To fully evaluate the effectiveness of a model, you must examine **both** precision and recall.
- Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa.

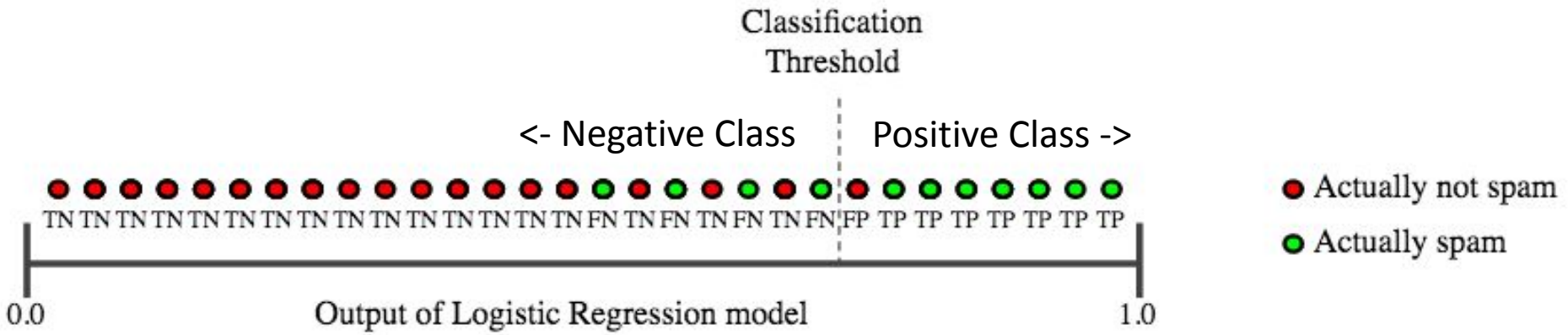
Example



True Positives (TP): 8	False Positives (FP): 2
False Negatives (FN): 3	True Negatives (TN): 17

- Precision = 0.8, Recall = 0.73
- Let’s increase the classification threshold

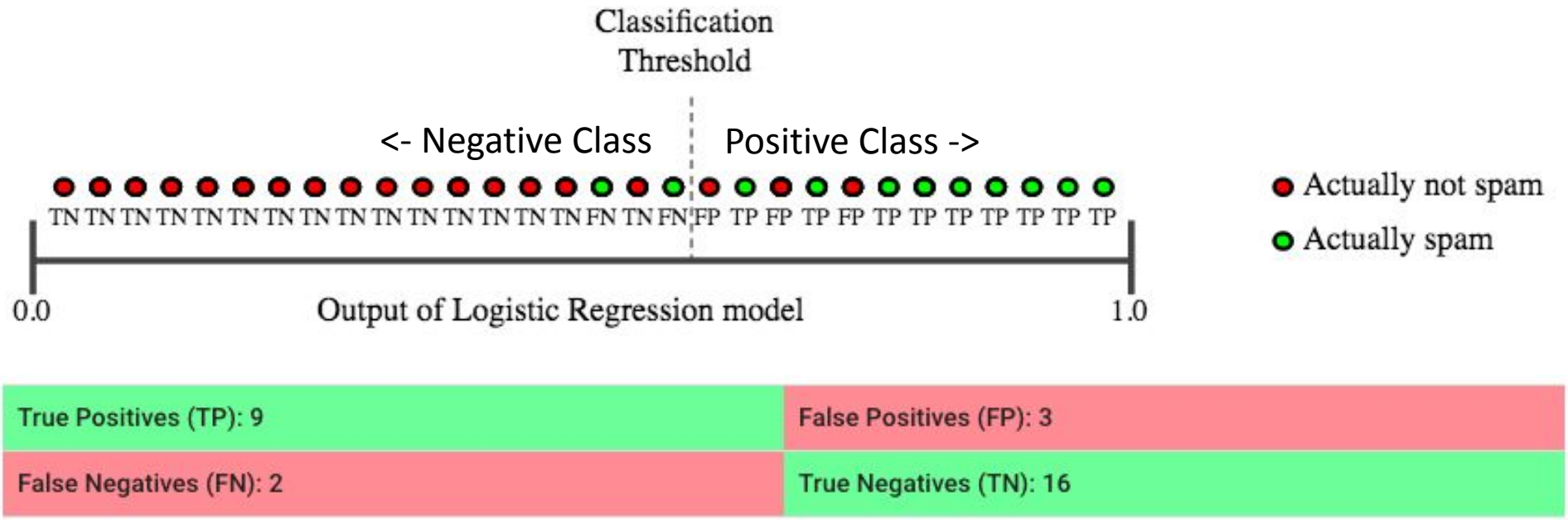
Increase the classification threshold



True Positives (TP): 7	False Positives (FP): 1
False Negatives (FN): 4	True Negatives (TN): 18

- Precision = 0.88, Recall = 0.64

Lower the classification threshold



True Positives (TP): 9	False Positives (FP): 3
False Negatives (FN): 2	True Negatives (TN): 16

- Precision = 0.75, Recall = 0.82

F1 Score

- The F1 Score is

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}$$

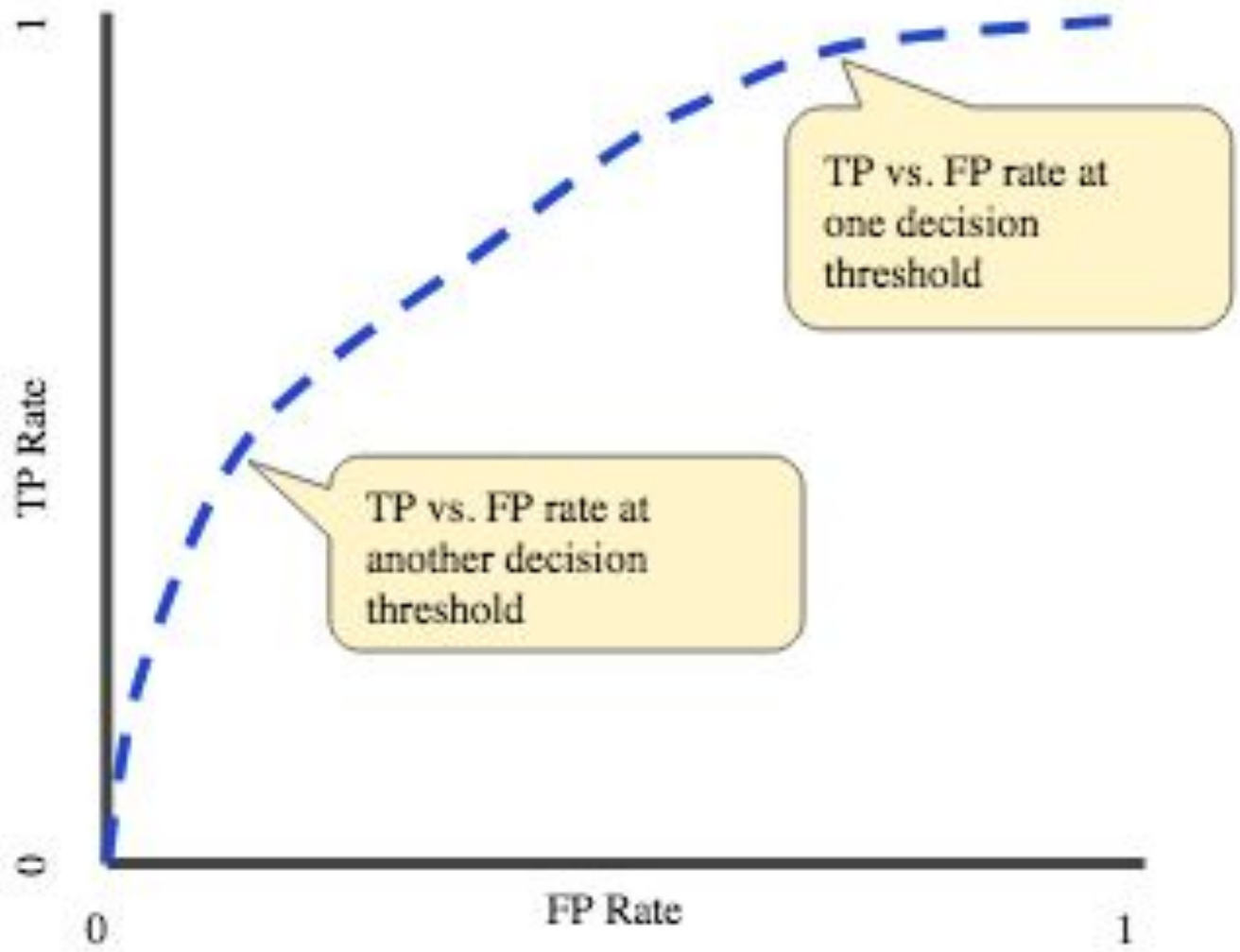
- It is also called the F Score or the F Measure.
- Put another way, the F1 score conveys the balance between the precision and the recall.
- If we were looking to select a model based on a balance between precision and recall, the model with the highest F1 score would win.

ROC

- The **Receiver Operating Curve** (ROC) plots TPR vs. FPR at different classification thresholds.
- Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

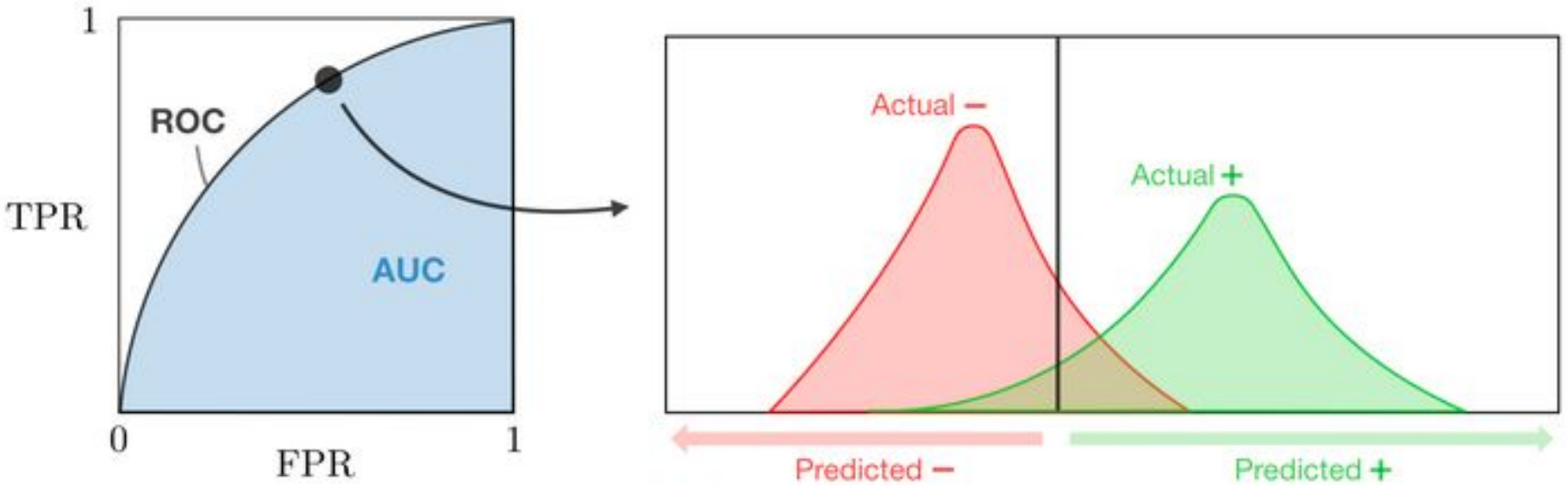
Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$	1-specificity

ROC


1
TP Rate
0
TP vs. FP rate at one decision threshold
TP vs. FP rate at another decision threshold
0
FP Rate
1

AUC

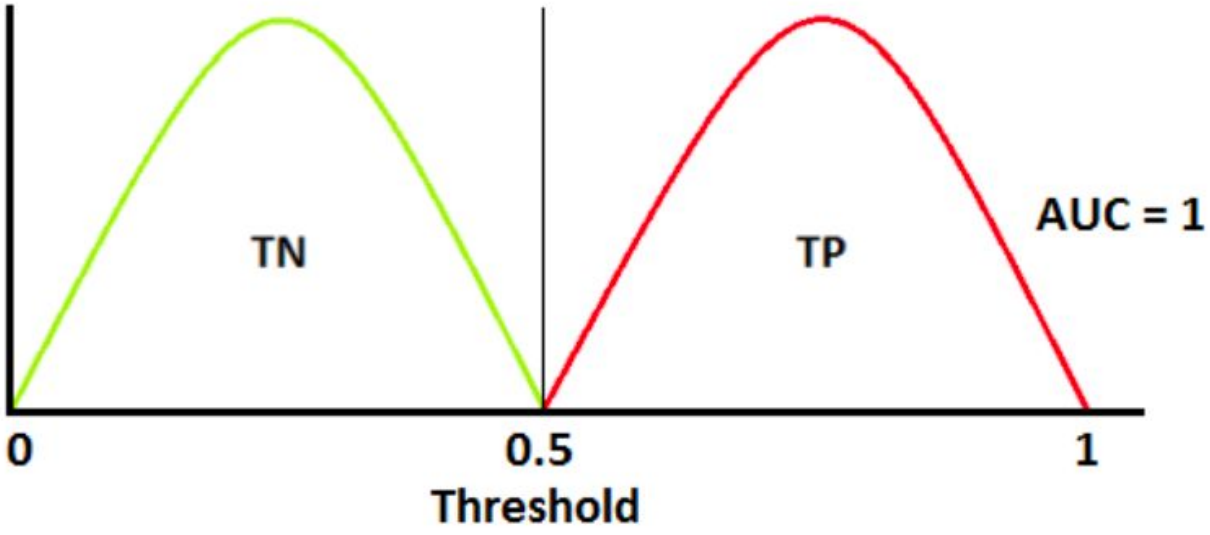
- The **area under** the receiving operating **curve**, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:

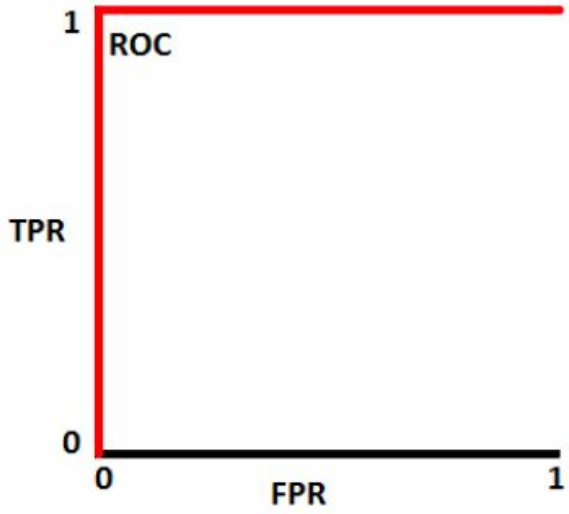


AUC Interpretation

- It's the probability that a classifier will be more confident that a randomly chosen positive example is actually positive than that a randomly chosen negative example is positive.
- It provides an aggregate measure of performance across all possible classification thresholds.
- AUC is a measure of seperability.
  - The higher the AUC, the better the model is at predicting class 0 as 0, class 1 as 1
  - AUC = 1 => model has ideal seperability. Predictions are 100% correct.
  - AUC = 0.5 => model has no seperability
  - AUC = 0 => model is reciprocating the results (0 as 1, 1 as 0). Predictions are 100% incorrect.

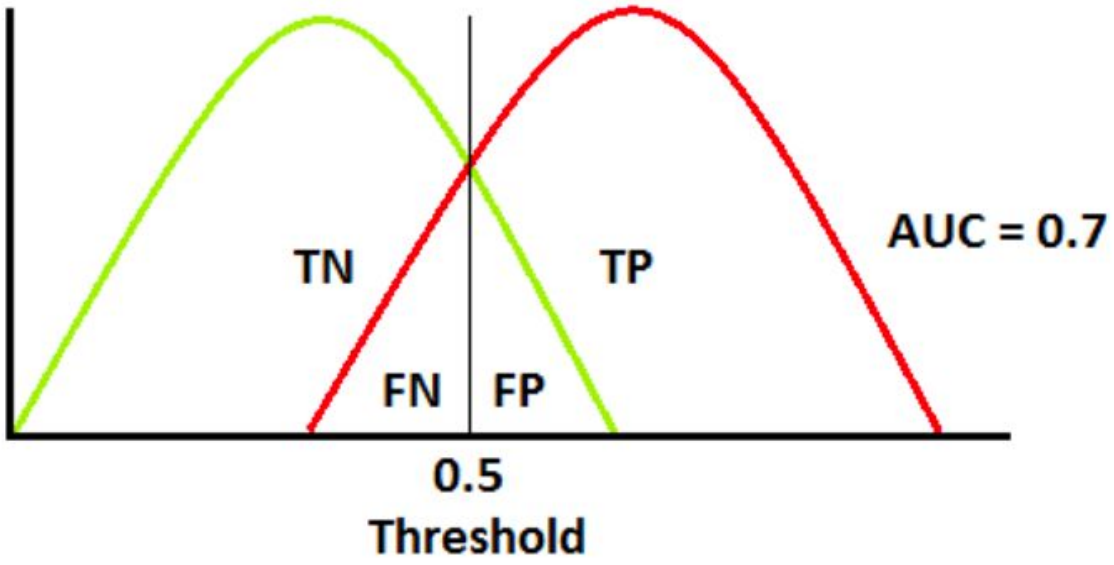
AUC = 1

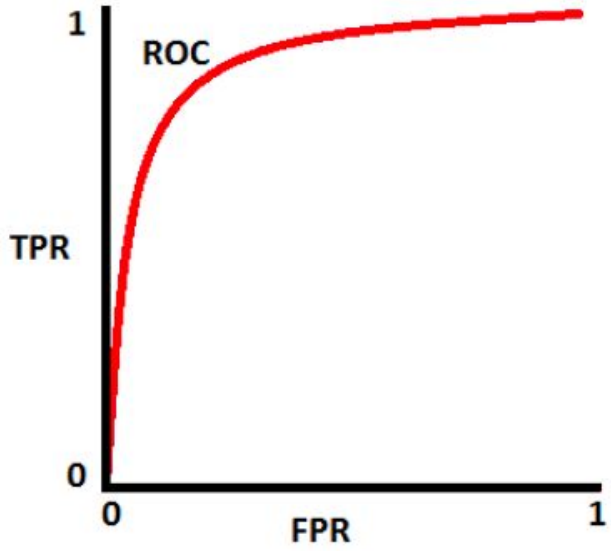




There is a 100% chance the model will be able to distinguish between the two classes.

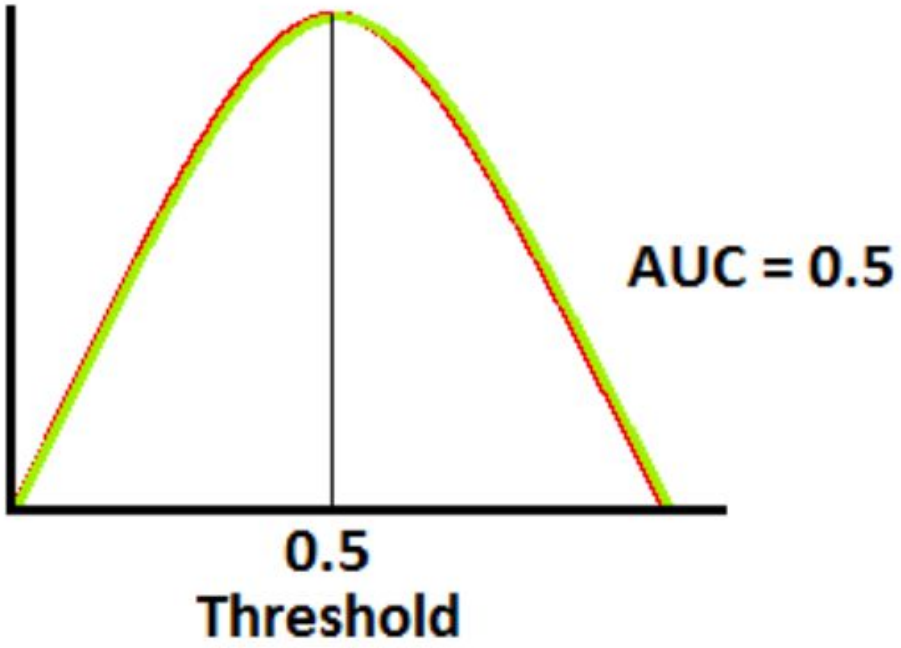
AUC = 0.7

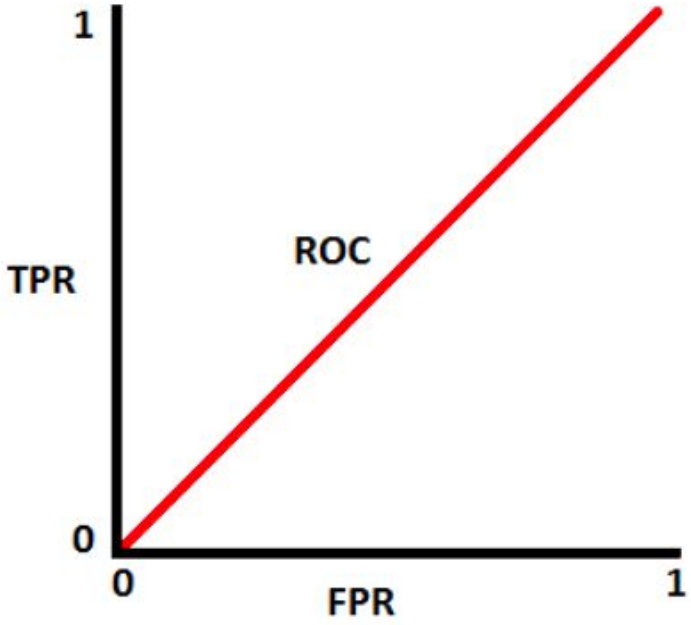




There is a 70% chance the model will be able to distinguish between the two classes.

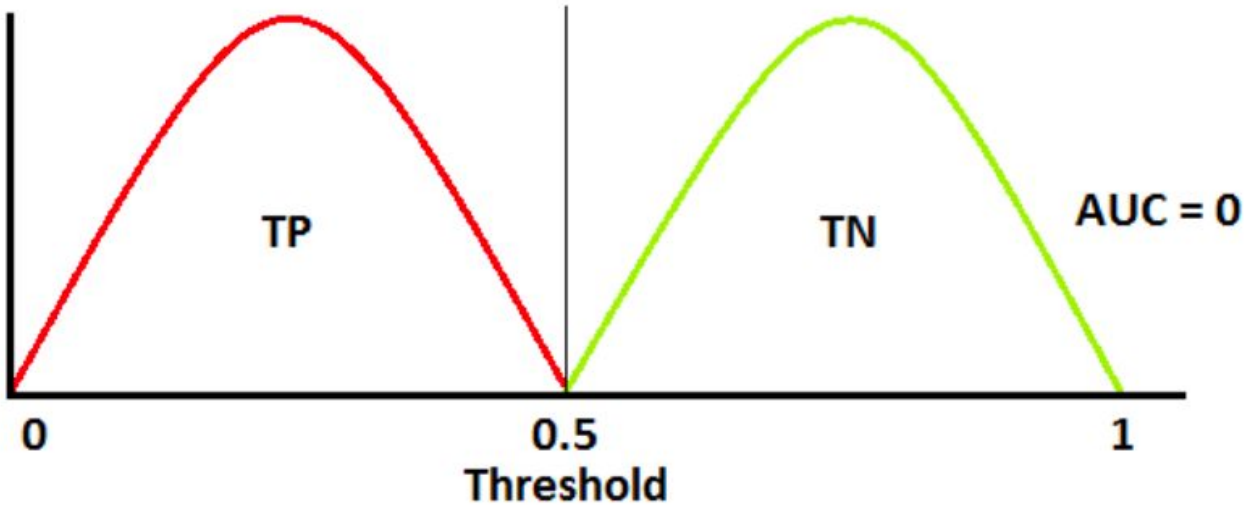
AUC = 0.5

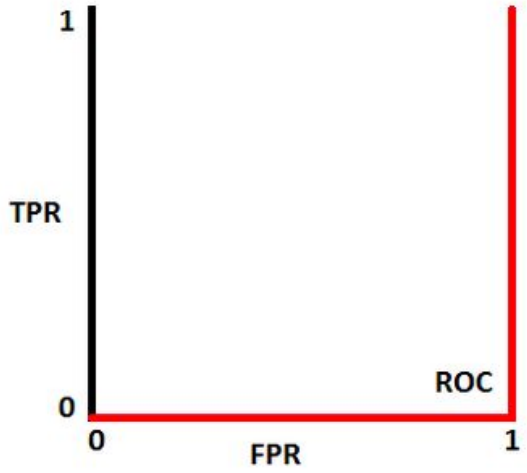




The model will not be able to distinguish between the two classes.

AUC = 0





The model is reciprocating the two classes.

Prediction Bias

- Logistic regression predictions should be unbiased. That is: *"average of predictions" ≈ "average of observations"*
- **Prediction bias** is a quantity that measures how far apart those two averages are.

$$\text{prediction bias} = \text{average of predictions} - \text{average of labels in data set}$$

- A significant nonzero prediction bias tells you there is a bug somewhere in your model, as it indicates that the model is wrong about how frequently positive labels occur.
- A zero prediction bias does not mean you have a great model.
  - A really terrible model could have a zero prediction bias.
  - For example, a model that just predicts the mean value for all examples would be a bad model, despite having zero bias.

Prediction Bias: Possible Root Causes

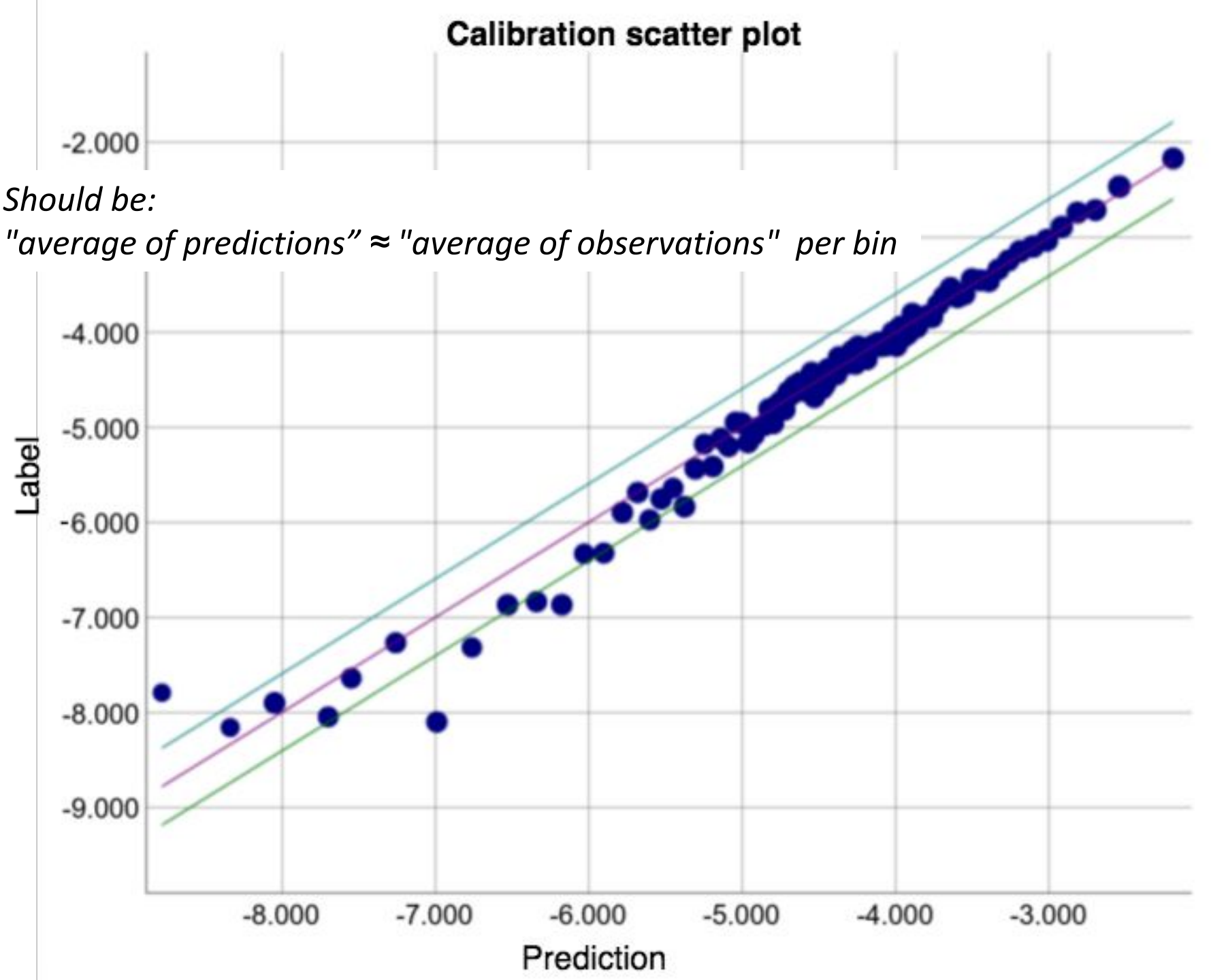
- Incomplete feature set
- Noisy data set
- Buggy pipeline
- Biased training sample
- Overly strong regularization

Prediction Bias: Possible Fixes

- Add a fudge factor? E.g. If your model has a +3% prediction bias, subtract 3% from model predictions.
  - **NO! Bad idea.**
  - You're fixing the symptom rather than the cause.
  - You've built a more brittle system that you must now keep up to date.

How to compute?

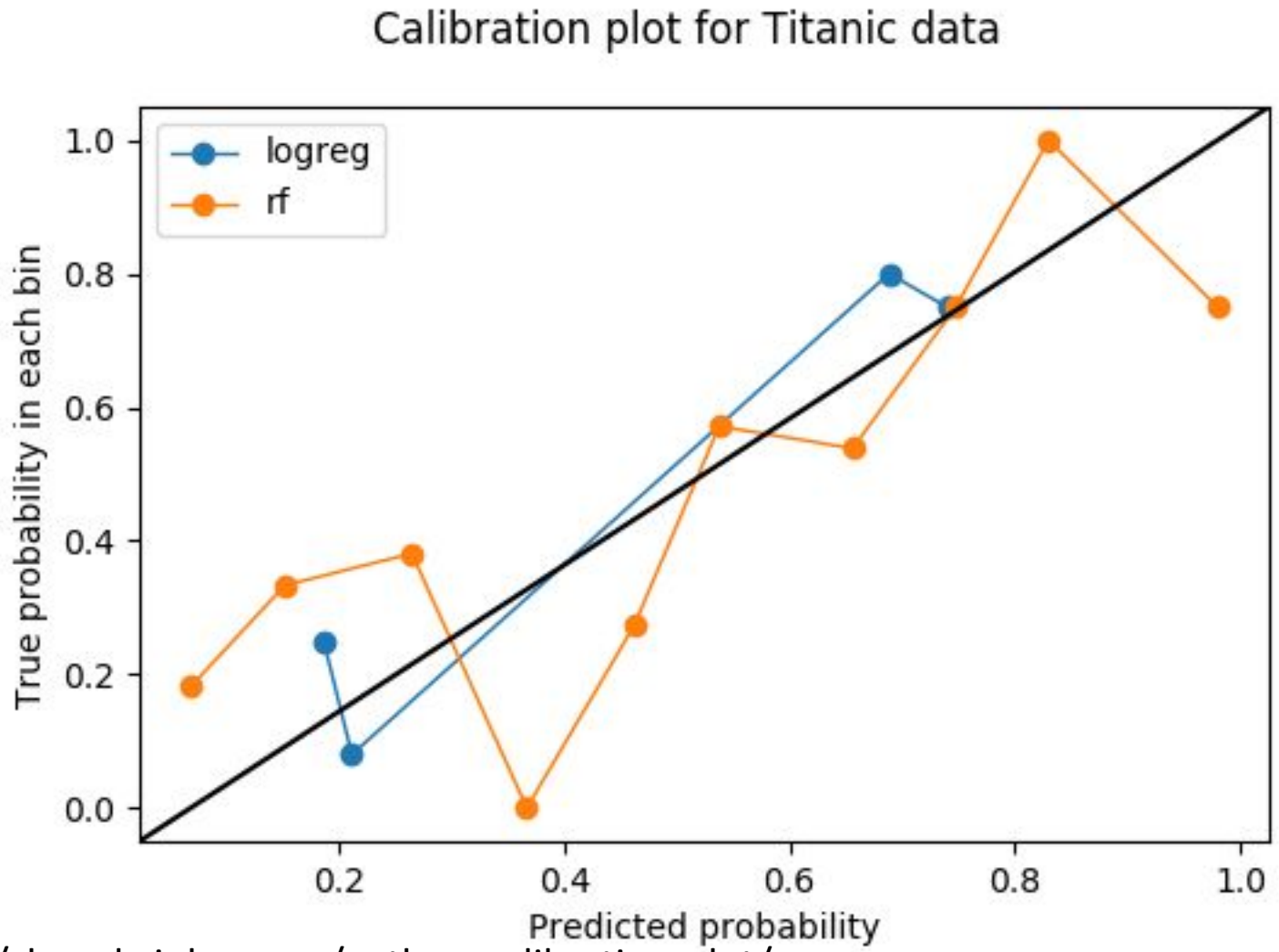
- Logistic regression output is a number between 0 and 1.
- Labeled examples are either 0 or 1.
- How do we calculate the average of labeled examples?
- *You cannot accurately determine the prediction bias based on only one example — you must examine the prediction bias on a bucket of examples.*



Should be:

"average of predictions" ≈ "average of observations" per bin

Calibration Plot



https://changhsinlee.com/python-calibration-plot/

Data set: https://www.kaggle.com/c/titanic

Calibration Plot Steps

- Create a data set with two columns that are actual label and its predicted probability given by the model.
- Sort this data set in ascending order of the probability predicted by the model.
- Now divide the data set in bins of some fixed size . If the data set is large then keep bin size large and vice versa.
- Now calculate fraction of actual positive in each bin and the average of probabilities predicted by the model.
- Plot a graph with fraction of positive on y-axis and average probability on x-axis.

Calibration Plot Implementation Examples

- Kaggle
- **A Guide to Calibration Plots in Python**
  - **Jupyter notebook**

MULTICLASS (MULTINOMIAL) LOGISTIC REGRESSION

Multiclass Logistic Regression

- Logistic regression is designed for two-class problems, modeling the target using a **binomial** probability distribution function.
  - The class labels are mapped to 1 for the positive class or outcome and 0 for the negative class or outcome. The fit model predicts the probability that an example belongs to class 1.
  - By default, logistic regression cannot be used for classification tasks that have more than two class labels, so-called multi-class classification.

Multiclass Logistic Regression

- Heuristic approaches
  - Split the multi-class classification problem into multiple binary classification problems and fit a standard logistic regression model on each subproblem.
  - Techniques of this type include one-vs-rest and one-vs-one wrapper models.
- Probabilistic approach
  - Multinomial Logistic Regression: Modified version of logistic regression that predicts a multinomial probability (i.e. more than two classes) for each input example.
  - AKA softmax regression

One-vs-Rest (OVR)

- For example, given a multi-class classification problem with examples for each class ‘red,’ ‘blue,’ and ‘green‘. This could be divided into three binary classification datasets as follows:
  - Binary Classification Problem 1: red vs [blue, green]
  - Binary Classification Problem 2: blue vs [red, green]
  - Binary Classification Problem 3: green vs [red, blue]
- This approach requires that each model predicts a class membership probability or a probability-like score. The argmax of these scores (class index with the largest score) is then used to predict a class.
  - The argmax function returns the argument or arguments (arg) for the target function that returns the maximum (max) value from the target function.
- A possible downside of this approach is that it requires one model to be created for each class.
  - For example, three classes requires three models. This could be an issue for large datasets (e.g. millions of rows), slow models (e.g. neural networks), or very large numbers of classes (e.g. hundreds of classes).
- Further reading:
  - https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

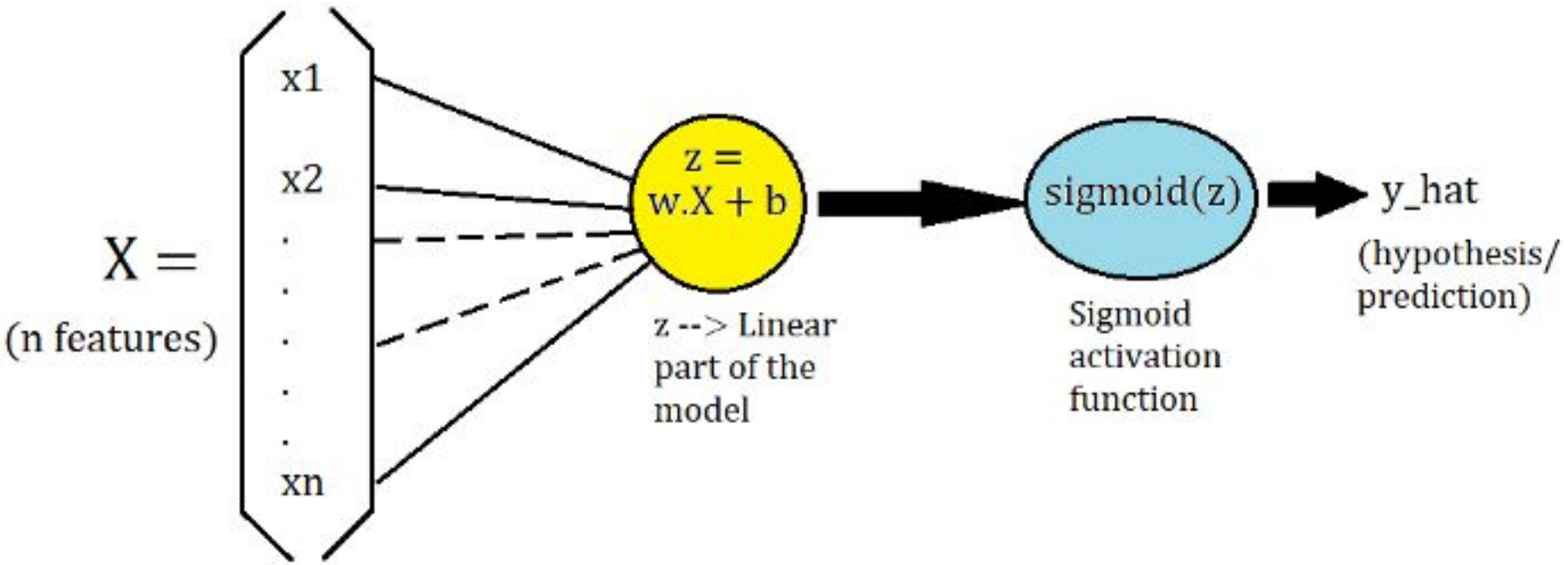
One-vs-One (OVO)

- For example, consider a multi-class classification problem with four classes: ‘red,’ ‘blue,’ and ‘green,’ ‘yellow.’ This could be divided into six binary classification datasets as follows:
  - Binary Classification Problem 1: red vs. blue
  - Binary Classification Problem 2: red vs. green
  - Binary Classification Problem 3: red vs. yellow
  - Binary Classification Problem 4: blue vs. green
  - Binary Classification Problem 5: blue vs. yellow
  - Binary Classification Problem 6: green vs. yellow
- Each binary classification model may predict one class label and the model with the most predictions or votes is predicted by the one-vs-one strategy.
- This is significantly more datasets, and in turn, models than the one-vs-rest strategy described in the previous section.
- Further reading:
  - https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

Multinomial Logistic Regression

- A logistic regression model that is adapted to learn and predict a multinomial probability distribution is referred to as Multinomial Logistic Regression.
- Similarly, we might refer to default or standard logistic regression as Binomial Logistic Regression.
- Changing logistic regression from binomial to multinomial probability requires a change to the loss function used to train the model (e.g. log loss to **cross-entropy loss**), and a change to the output from a single probability value to one probability for each class label.

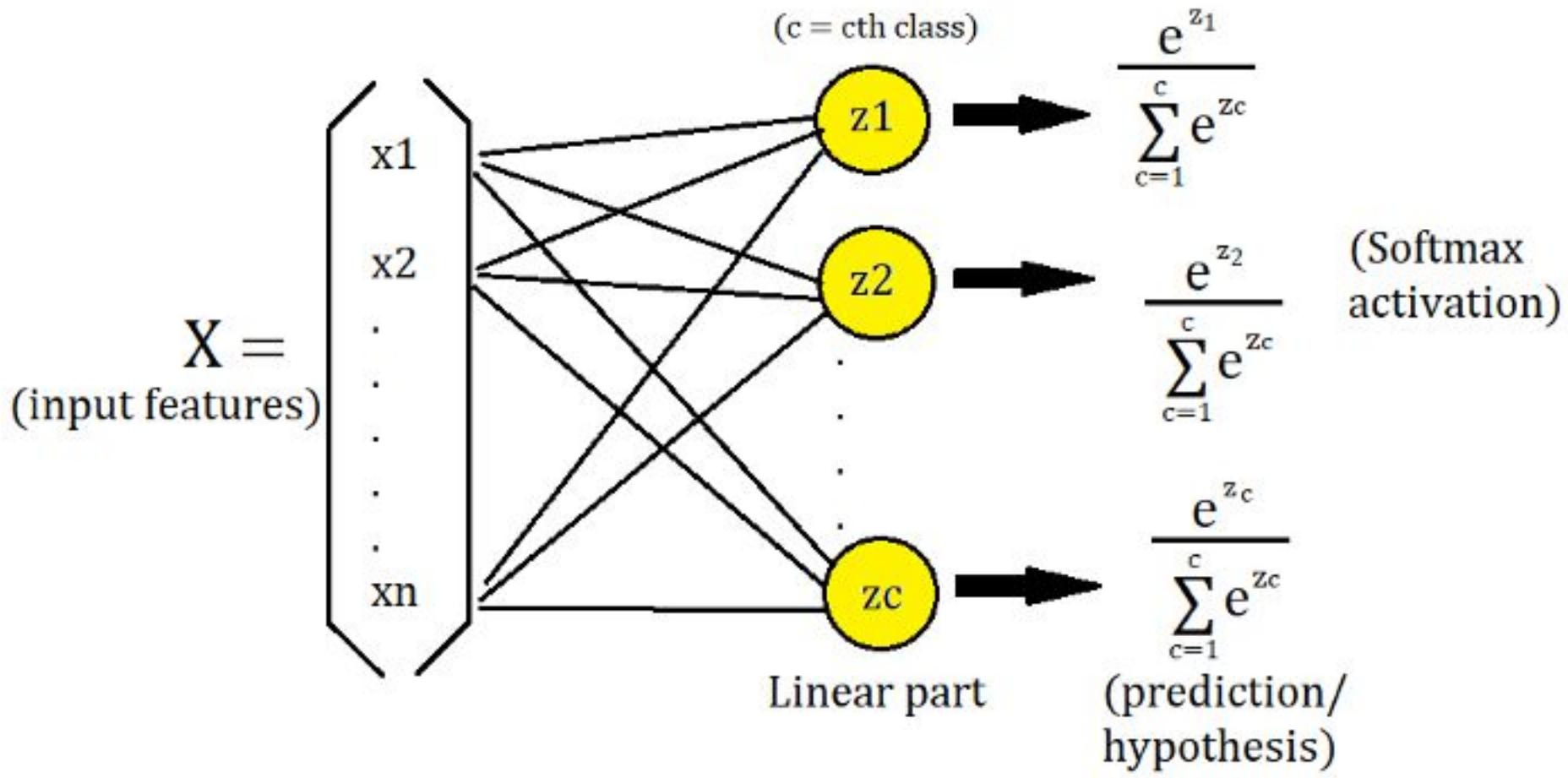
Logistic Regression w/ sigmoid



$$z = b + w_1 x_1 + w_2 x_2 + \ldots + w_N x_N$$

$$y' = \frac{1}{1 + e^{-z}}$$

Multiclass Logistic Regression w/ Softmax



Multinomial Logistic Regression Further Reading

- https://machinelearningmastery.com/multinomial-logistic-regression-with-python/
- https://towardsdatascience.com/softmax-regression-in-python-multi-class-classification-3cb560d90cb2
- http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/
- https://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html