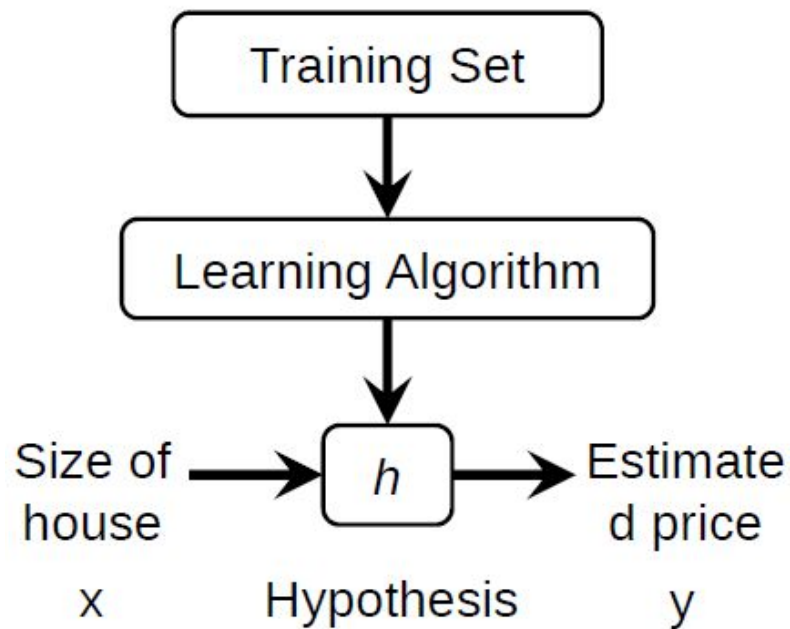


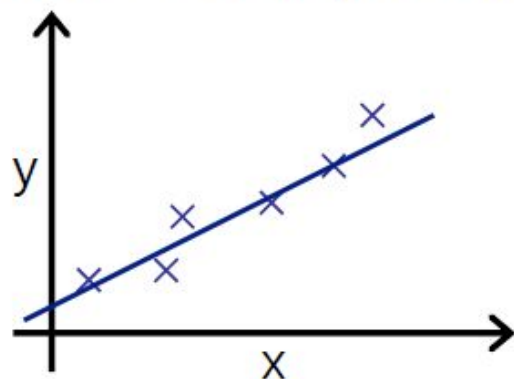
# Text Classification II

Logistic Regression

# Quick Review on Linear Regression



How do we represent  $h$  ?

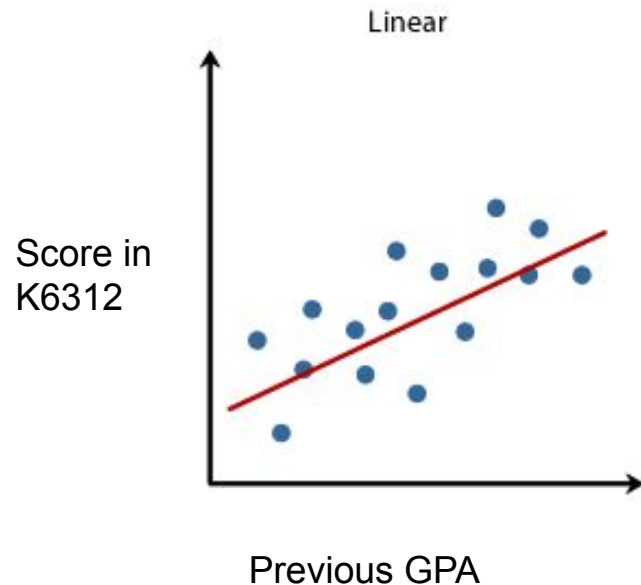


$$h(x) = w_0 + w_1x$$

Linear regression with one variable.  
**“Univariate Linear Regression”**

# Continuous Target

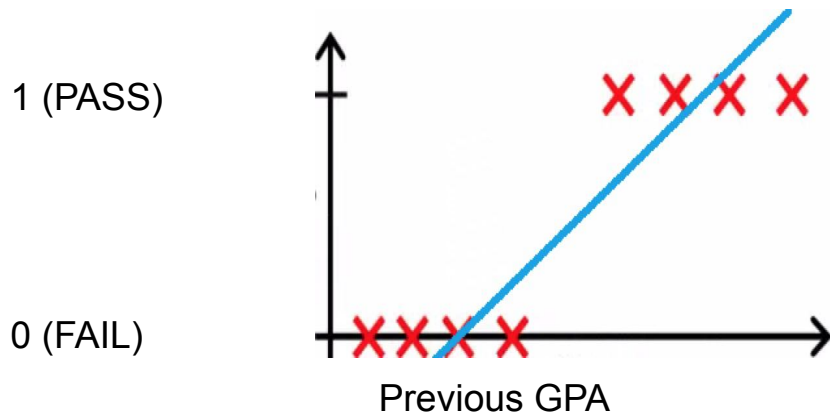
- Let us build an auto-grade algorithms
- Input feature is one scalar: your previous GPA
- Target value is: your score in K6312



# Discrete Target

- We only want to predict whether you can pass K6312
- Input feature is one scalar: your previous GPA
- Target value is: a binary value (1: pass, 0: fail)

## Classification Problem



# Classification

## Binary Classification

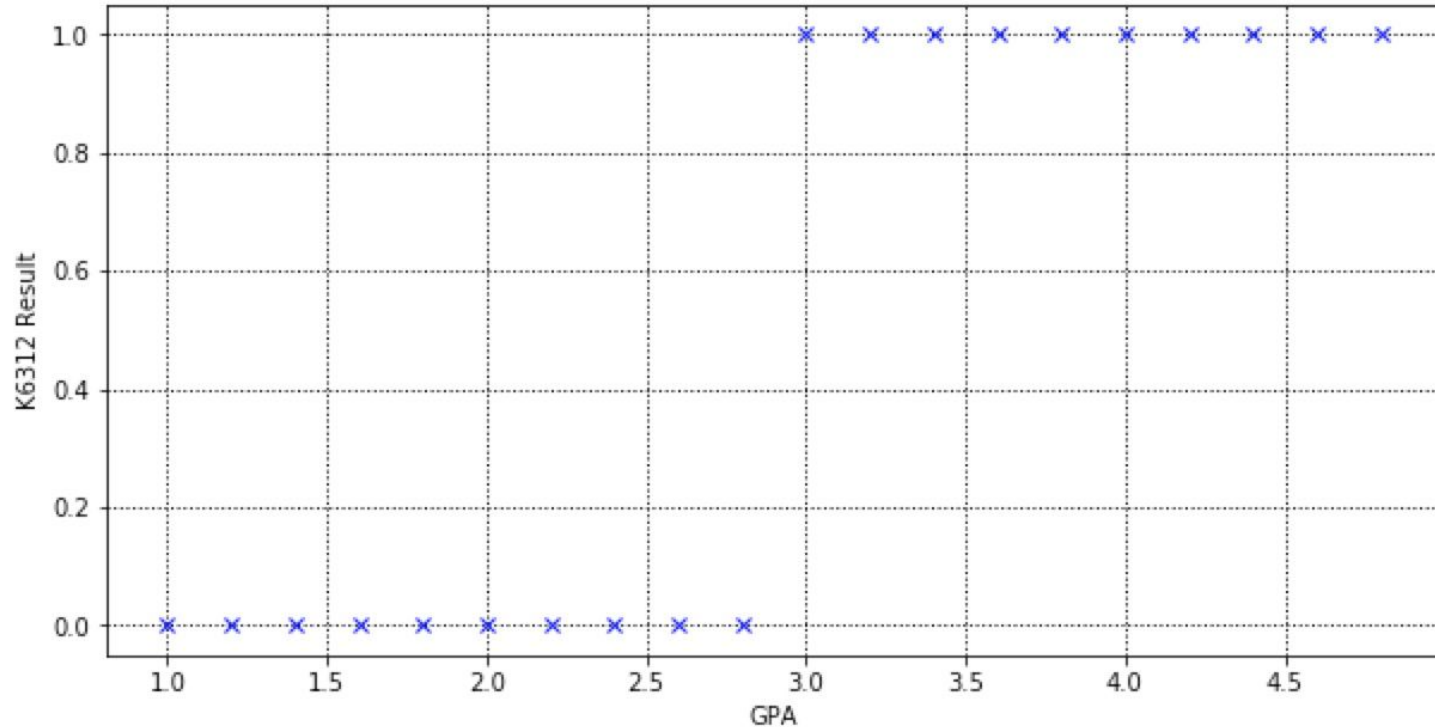
- Email: spam or not spam
- Online Transaction: fraud or not fraud
- .....

$$y = \begin{cases} -1 & \text{Negative class, e. g. not spam and not fraud} \\ 1 & \text{Positive class, e. g. spam and fraud} \end{cases}$$

Machine Learning is to learn a function from data such that

$$f: X \rightarrow Y$$

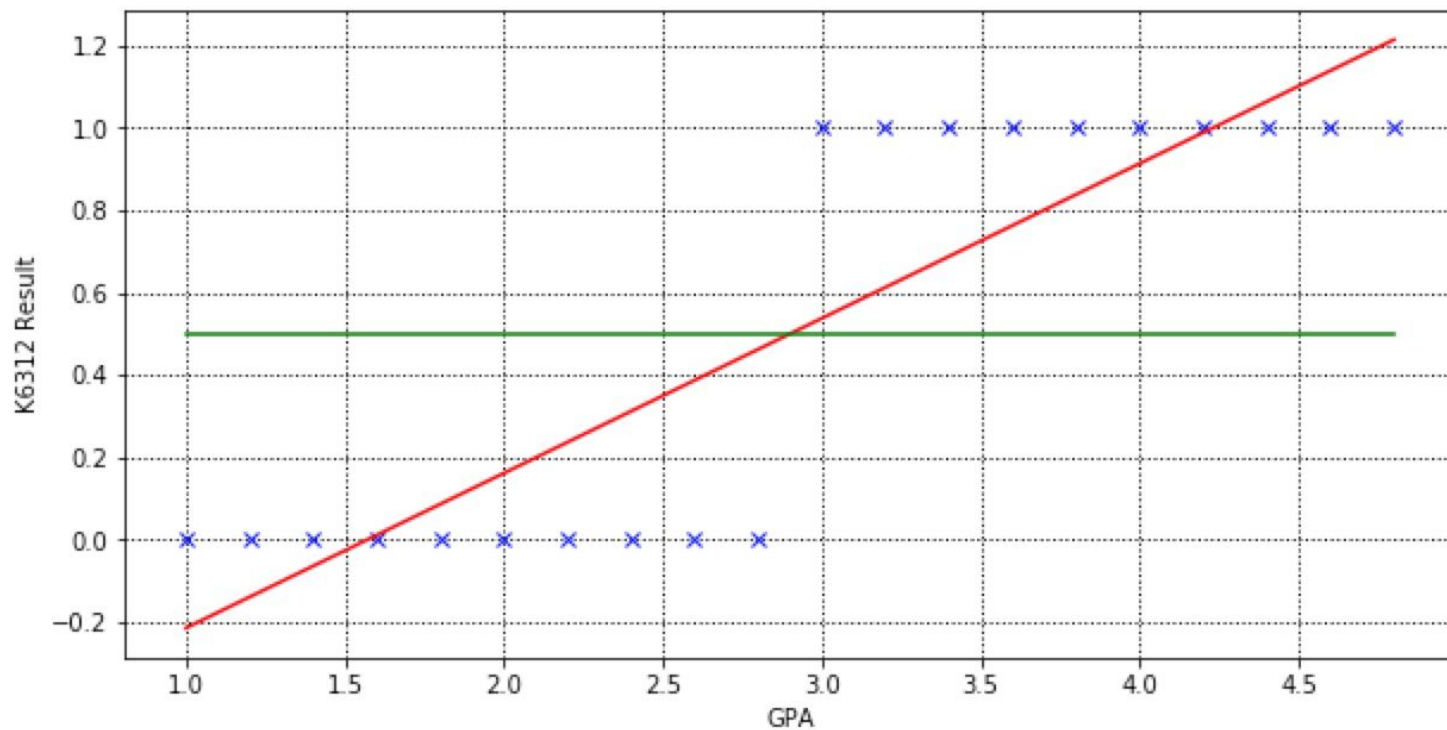
# Can we use linear regression for classification?



After fitting,

```
print(lin_regression.coef_)  
print(lin_regression.intercept_)
```

```
[ 0.37593985]  
-0.5902255639097744
```





# Output Value is continuous

- For classification problem, we want the output value to be probabilistic, which should be in range(0, 1).
- However, the output of linear regression is unbounded

```
print(lin_regression.coef_)  
print(lin_regression.intercept_)
```

```
[ 0.37593985]  
-0.5902255639097744
```

$$y=0.3759*x-0.59$$

**When  $x = 5$ ,  $y=1.289$**

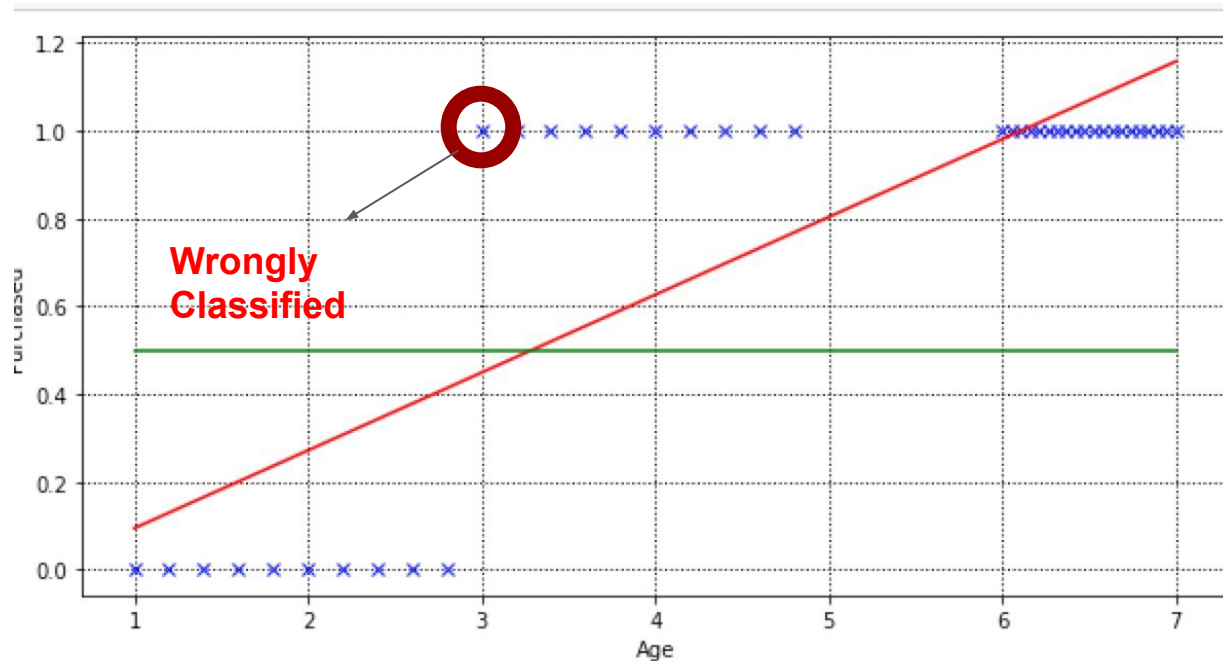
**When  $x = 4.6$ ,  $y=1.038$**

**When  $x = 1.2$ ,  $y=-0.13$**

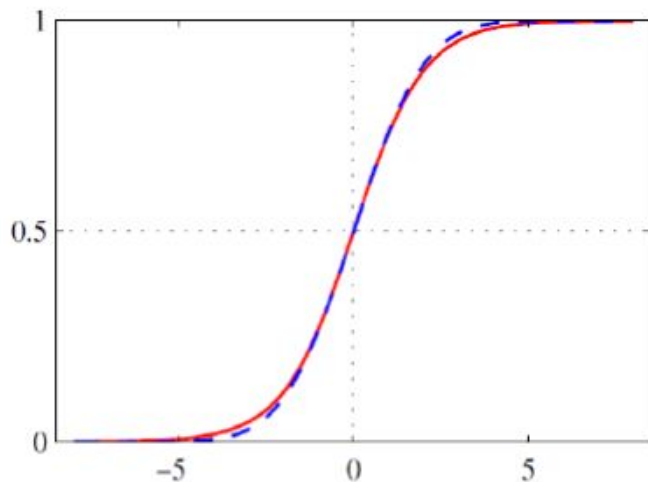
What we want is that the proba. Scores of the first two cases is close to 1 and the last case is close to 0.

# Imbalanced data

- Let us add 20 students whose GPA are in the range(6, 7) and pass the K6312



# Logistic Function



$t$  : (-infinite, +infinite)  
 $\sigma(t)$ : probabilistic score from 0 to 1

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

Linear  
Regression



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

Logistic  
Regression

$$f(x) = w * x + b$$

$$f(x) = \frac{1}{1+e^{-(w*x+b)}}$$

# Logistic Regression

# Logistic Regression

- Uses logistic function to model binary target
- Model the distribution of  $p(y = 1|\mathbf{x})$  given  $\mathbf{x}$
- The exact parametric formulation is:

$$p(y = 1|x) = \frac{1}{e^{(-wx+b)} + 1} = \frac{e^{(wx+b)}}{e^{(wx+b)} + 1}$$

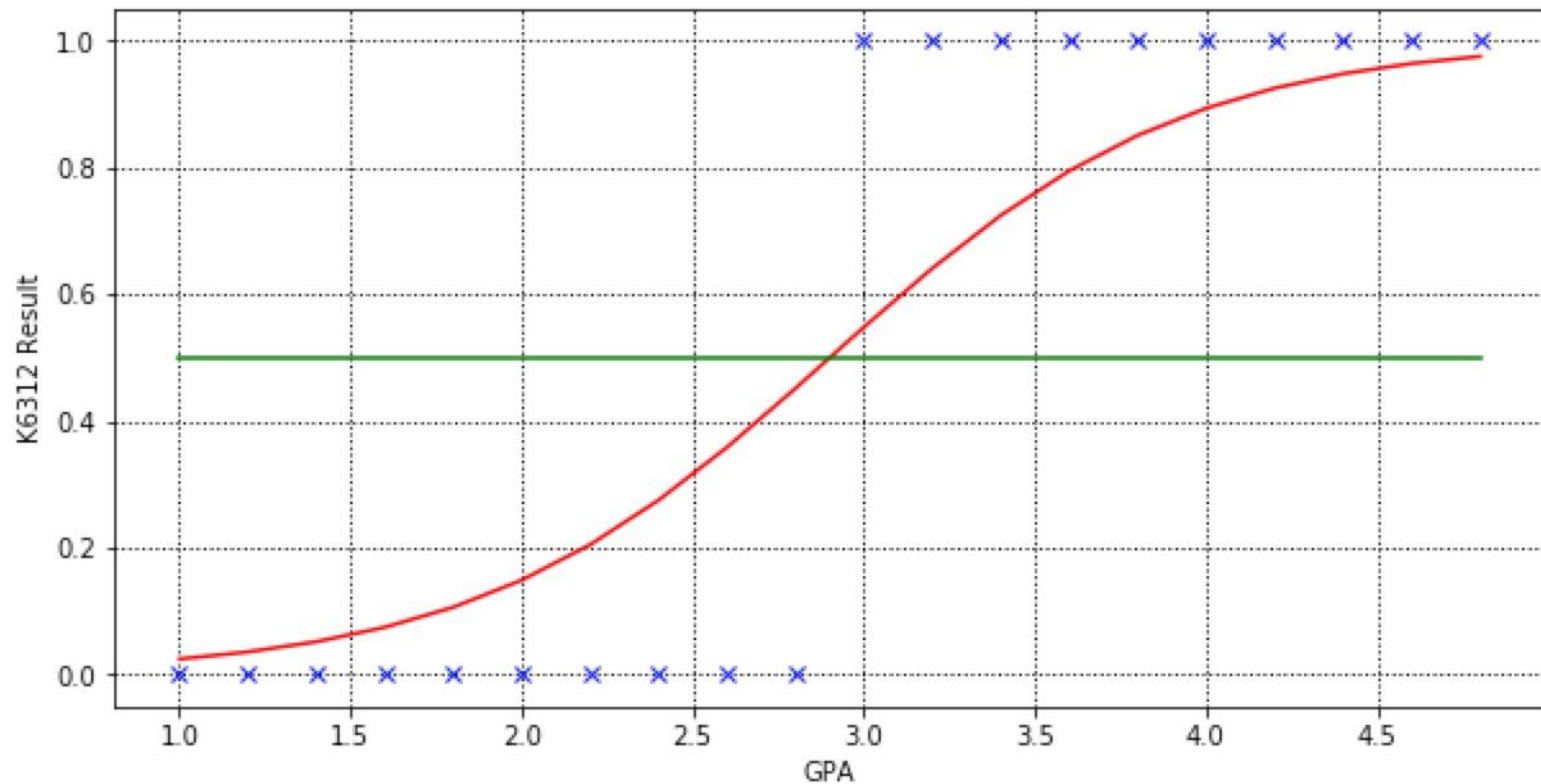
$$p(y = 0|x) = 1 - \frac{1}{e^{(-wx+b)} + 1} = \frac{1}{e^{(wx+b)} + 1}$$

- Let us check the performance of Logistic Regression on K6312 auto-grade system

# After fitting

```
print(log_regression.coef_)  
print(log_regression.intercept_)
```

```
[[1.93582432]]  
[-5.61388646]
```



# Output Value is Prob.score

```
print(log_regression.coef_)  
print(log_regression.intercept_)
```

```
[[1.93582432]]  
[-5.61388646]
```

$$p(y = 1|x) = \frac{1}{e^{(-wx+b)} + 1} = \frac{e^{(wx+b)}}{e^{(wx+b)} + 1}$$

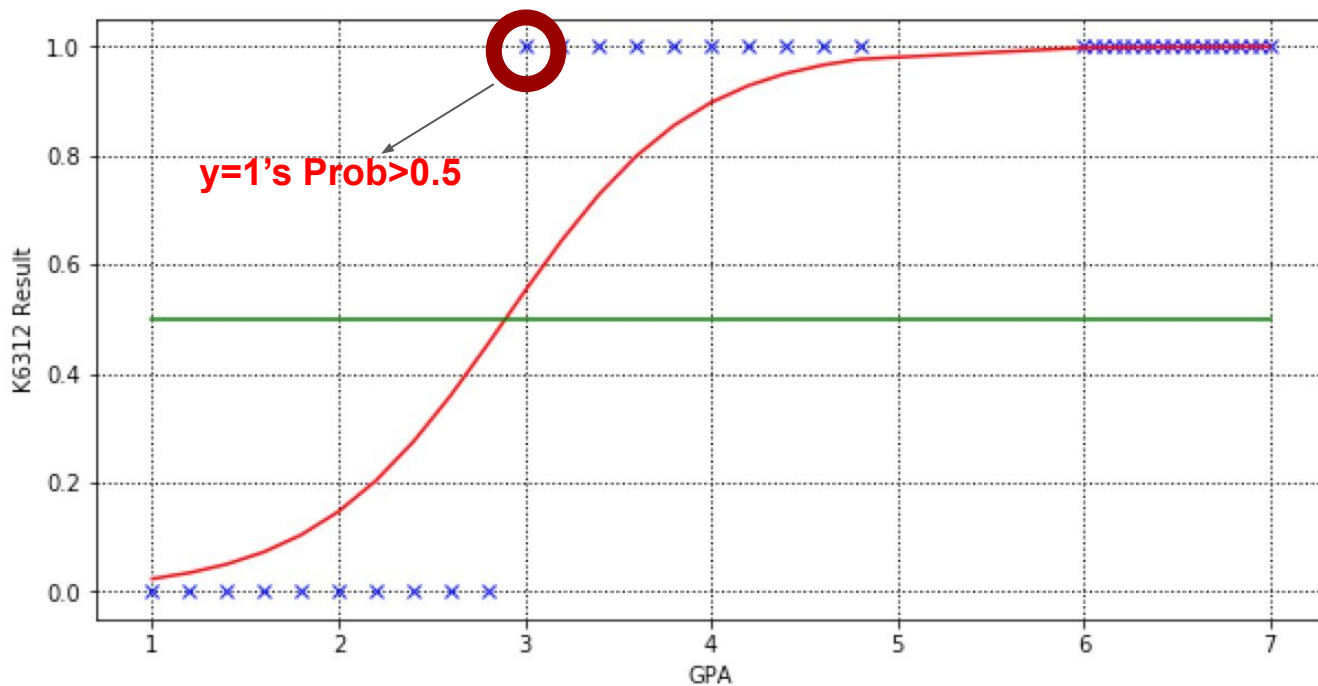
$$\begin{aligned} p(y=1|x) &= e^t / (1 + e^t) \\ p(y=0|x) &= 1 / (1 + e^t) \\ t &= 1.93 * x - 5.6 \end{aligned}$$

	prob(y=0)	prob(y=1)
<b>x = 5</b>	[0.01686949	0.98313051]
<b>x = 4.6</b>	[0.03588451	0.96411549]
<b>x = 1.2</b>	[0.96411521	0.03588479]]



# Imbalanced data

- Let us add 20 students whose GPA are in the range(6, 7) and pass the K6312



# How to learn parameters

- Fitting the data
- In the python code, it is simple

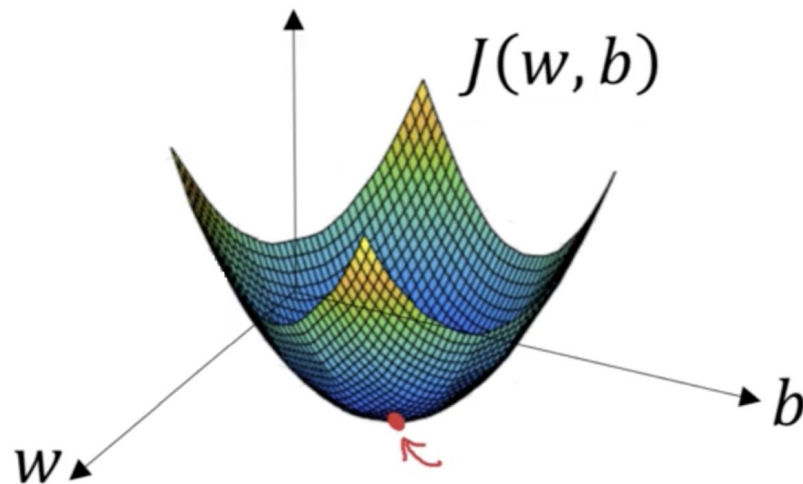
## Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
```

- Actually, it is an optimization problem (in math perspective)

# Optimization

- Fitting the data -> define a loss function, which reflects the fitness of the different model parameters over the parameters
- Optimization is the process to search the minimum point



# Math Warning

# Entropy Loss

- For each single data point:  $\tilde{y} = p(y = 1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$

$$Loss(y, \tilde{y}) = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

- To understand this loss function, compute the loss values for these following cases:
  - If  $y=1$ , predict prob of  $y=1$  is 0.9,
  - If  $y=0$ , predict prob of  $y=1$  is 0.2,
  - If  $y=1$ , predict prob of  $y=1$  is 0.2,
  - If  $y=0$ , predict prob of  $y=1$  is 0.9,

Whether the model  
prediction is good  
nor not?

# Entropy Loss

- For each single data point:  $\tilde{y} = p(y = 1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$

$$Loss(y, \tilde{y}) = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

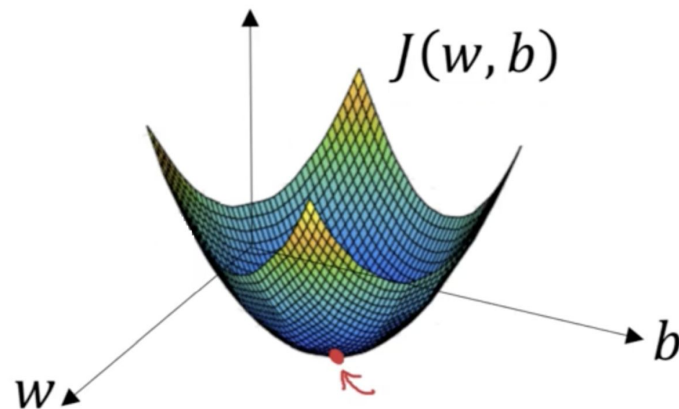
- To understand this loss function, compute the loss values for these following cases:
  - If  $y=1$ , predict prob of  $y=1$  is 0.9,  $-\log 0.9=0.1$  Good Fitness, Low Loss
  - If  $y=0$ , predict prob of  $y=1$  is 0.2,  $-\log 0.8=0.22$  Loss
  - If  $y=1$ , predict prob of  $y=1$  is 0.2,  $-\log 0.2=1.6$  Bad Fitness, High Loss
  - If  $y=0$ , predict prob of  $y=1$  is 0.9,  $-\log 0.1=2.3$  Loss

# Optimization is to reduce Loss

- For training data of  $m$  data points  $(x, y)$ , the loss is the function of model parameters

$$J(w, b) = \frac{\sum_{i=1}^m \text{Loss}(\tilde{y}^i, y^i)}{m}$$

$$\tilde{y} = p(y = 1|x) = \frac{1}{e^{(-wx+b)} + 1} = \frac{e^{(wx+b)}}{e^{(wx+b)} + 1}$$

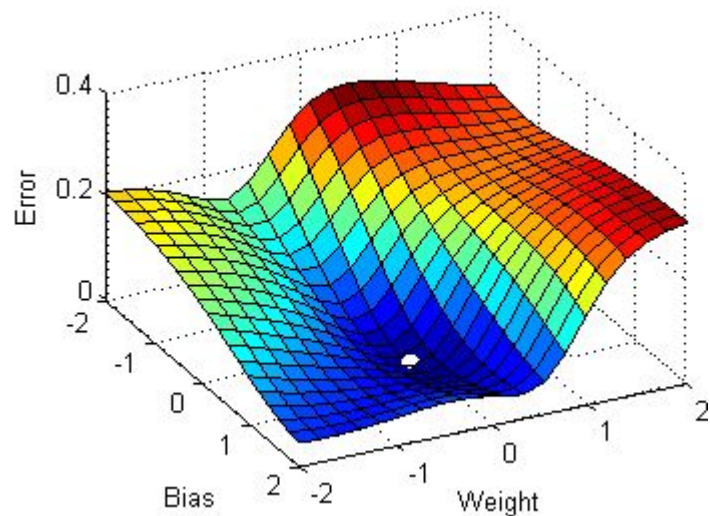


# Gradient Descent Algorithm

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \alpha \nabla J(\mathbf{w})$$

Diagram illustrating the Gradient Descent Algorithm equation:

- $\mathbf{w}_{n+1}$ : New Parameters Guess
- $\mathbf{w}_n$ : Current Parameters Guess
- $\alpha$ : Learning Rate
- $\nabla J(\mathbf{w})$ : Gradient for loss function  $J$  for  $\mathbf{w}$ , which is computed by BP algorithm



Like hiking down a mountain



# Decision boundary of Logistic Regression

- Decision is made by comparing the probabilities

$$p(y = 1|\mathbf{x}) > p(y = -1|\mathbf{x}) \Leftrightarrow \frac{p(y = 1|\mathbf{x})}{p(y = -1|\mathbf{x})} > 1$$

- Take the logarithm

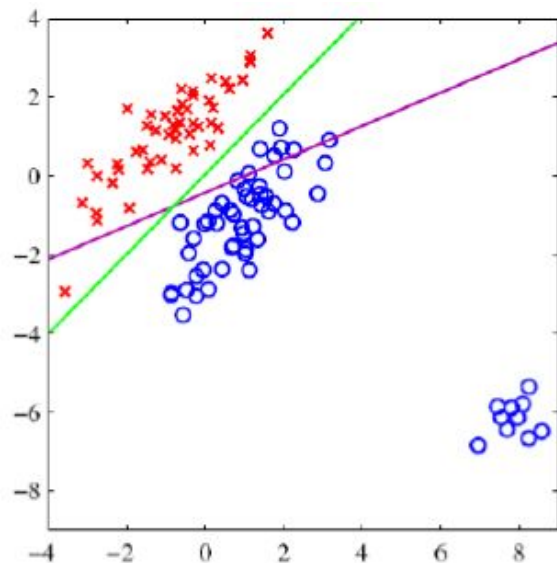
$$\ln \frac{p(y = 1|\mathbf{x})}{p(y = -1|\mathbf{x})} = \mathbf{w}^\top \mathbf{x} + b \rightarrow \mathbf{w}^\top \mathbf{x}$$

- Decision boundary is linear

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

$$y = \begin{cases} +1 & \text{if } \mathbf{w}^\top \mathbf{x} + b > 0 \\ -1 & \text{otherwise} \end{cases}$$

\* The threshold is tunable



# We can have multiple w

- For simplicity, we only have one feature x therefore only one w and bias b in the example.
- In practice, each data sample is represented by a **n-dimensional vector** and the logistic regression model has **n weights** and **one bias b**.
- For text mining, the input vectors will be BoW vectors.

output:  $\sigma(-1.2*(10) + 1.4*(5) + 2.2*(3) + 0.6*(5) + 0.2)$

# Evaluation

How to do we evaluate the model performance?

i.e., how to quantify the matching degree  
between the ground truth  $y$  and the predicted  
labels  $y^\wedge$ .

# Evaluation of Classification Problems

- Confusion Matrix

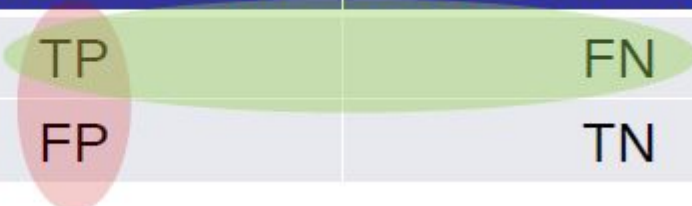
	Predicted Positive	Predicted Negative
Positive Label	<b>TP</b> True Positive	<b>FN</b> False Negative
Negative Label	<b>FP</b> False Positive	<b>TN</b> True Negative

- Accuracy: How accurate is the prediction?

$$\frac{\text{Correct Prediction}}{\text{Total \#-of-Samples}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

# Precision and Recall

	Predicted Positive	Predicted Negative
Positive Label	TP	FN
Negative Label	FP	TN



- Precision =  $\frac{TP}{TP + FP}$ 
  - how accurate the positive prediction is?

- Recall =  $\frac{TP}{TP + FN}$ 
  - how many positive cases are detected?

## Example: K6312 Auto-grade

	Predicted Positive	Predicted Negative
Positive Label	27	4
Negative Label	1	18

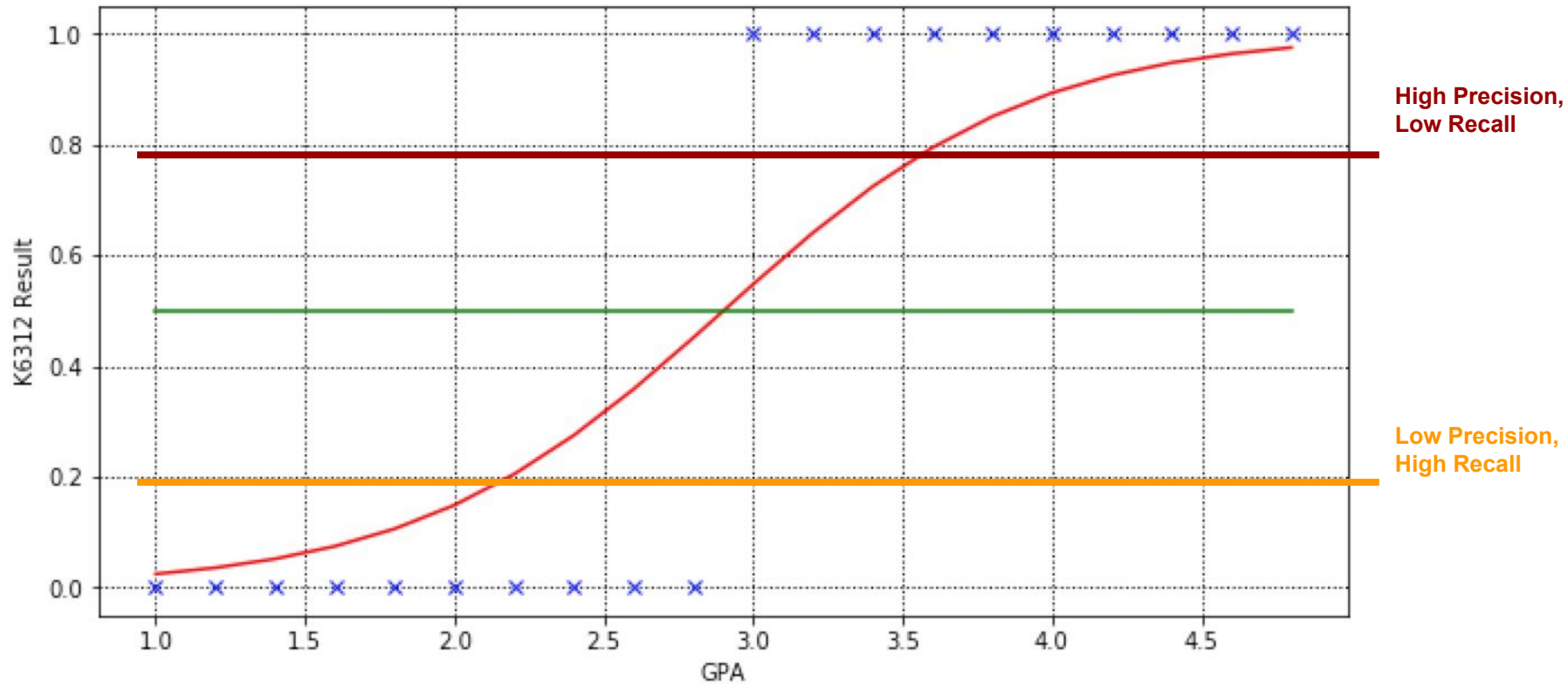
- Accuracy =  $(27 + 18) / (27 + 1 + 18 + 4) = 0.9$
- Precision =  $27 / (27 + 1) = 0.964$
- Recall =  $27 / (27 + 4) = 0.871$
  
- Can we do better?

# Precision vs Recall

- Case 1: Accuracy is high, but recall is low.
  - Examples?
- Case 2: Accuracy is high, but precision is low.
  - Examples?

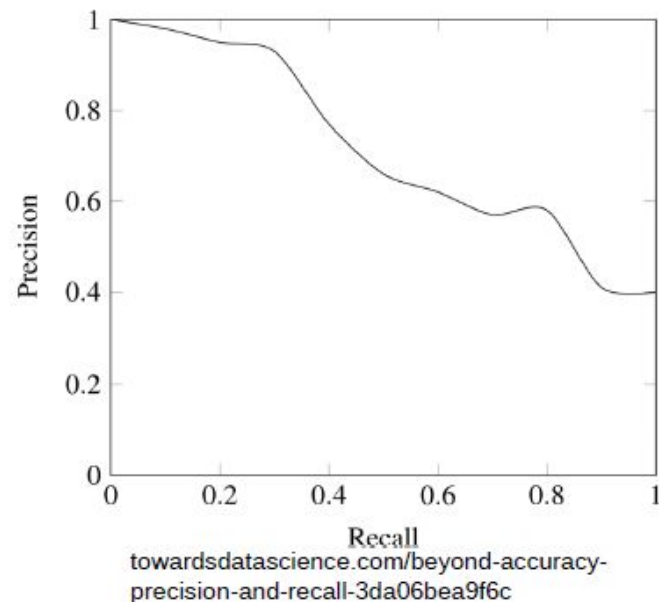


# Precision vs Recall



# Precision v.s. Recall

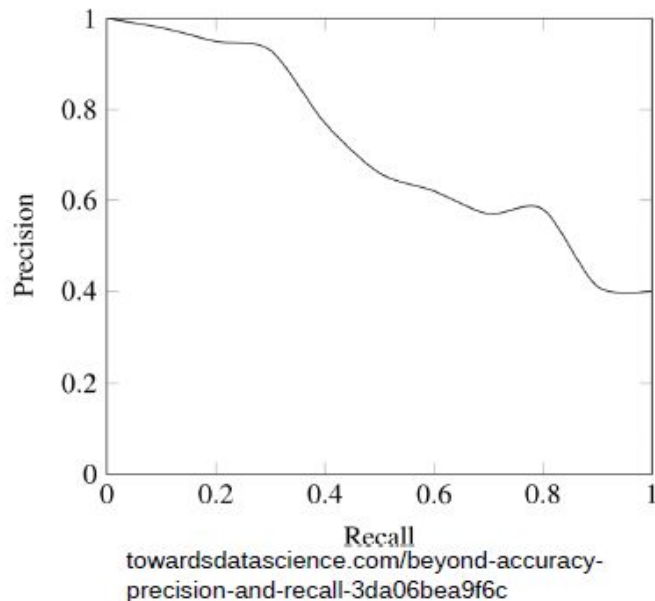
- Under non-trivial situation, precision and recall cannot be optimized at the same time
  - Which one to optimize depends on use cases



# F1 Score Vs Accuracy

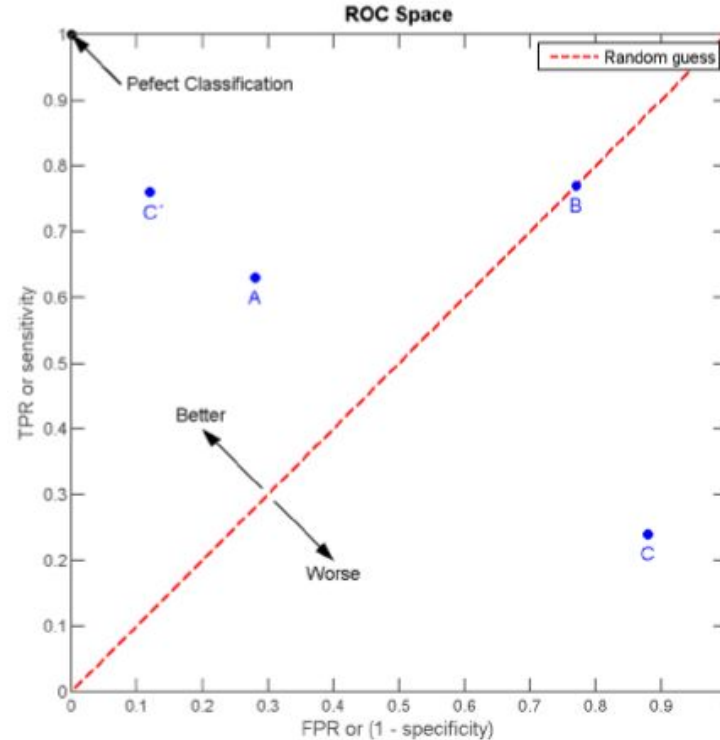
- Accuracy does not perform well for imbalanced data sets
  - Assume we have 100 transaction, 90 are non-fraud cases and 10 fraud ones
  - High accuracy can be achieved by classifying every transaction as non-fraud
- Precision and Recall can give more insights
- F1 Score conveys the balance between the precision and recall.

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$



# Receiver Operation Characteristic

- Illustrates the diagnostic ability of a binary classifier as its discrimination threshold varies
- True positive rate (TPR) against the false positive rate (FPR) at various threshold



# Receiver Operation Characteristic

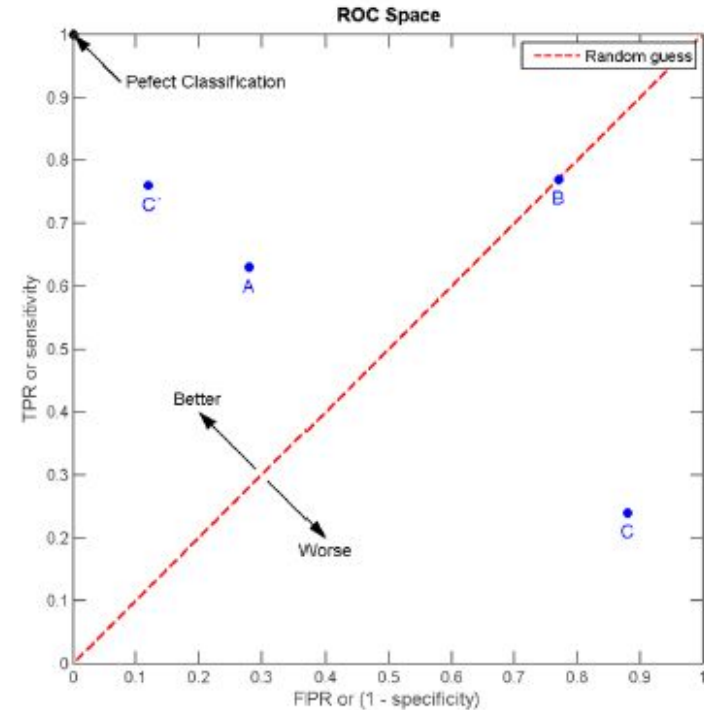
- **FPR**: Probability of False Alarm

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

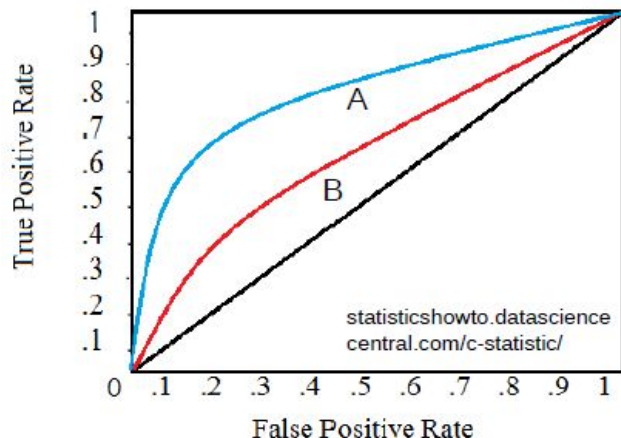
- **TPR**: Probability of Detection

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- The best possible prediction method would yield a point in the upper left corner (0,1)
- The (0,1) point is also called a **perfect classification**
- A random guess would give a point along a diagonal



# Area Under the Curve



Area Under Curve (AUC) is the area under the ROC curve

$$\text{AUC}(A) > \text{AUC}(B)$$

Classifier A is better than  
Classifier B

- ROC curve plots parametrically TPR(T) versus FPR(T) with threshold T as the varying parameter
- AUC equals to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example
- AUC is one of the most widely used metrics for evaluation of binary classification problem

# Multiclass Classification

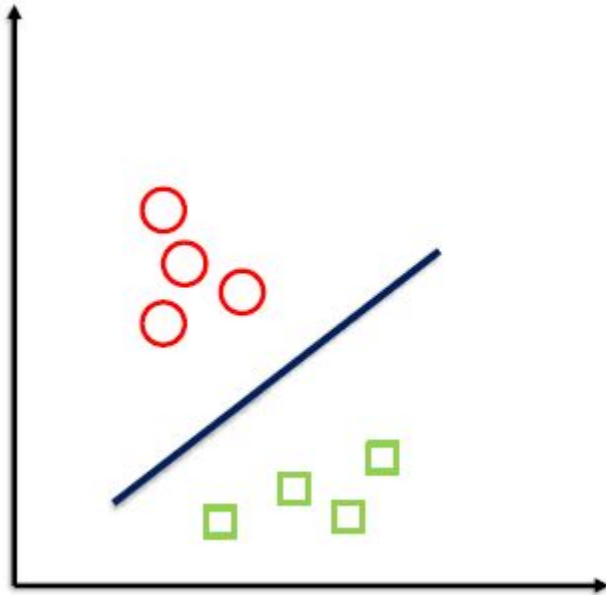
# Multiclass Classification

- Weather: Cloudy, Rain, Snow, ...
- Fruit: Apple, Orange, Peach, ...
- Email tagging: Work, Ad, Friends, ...

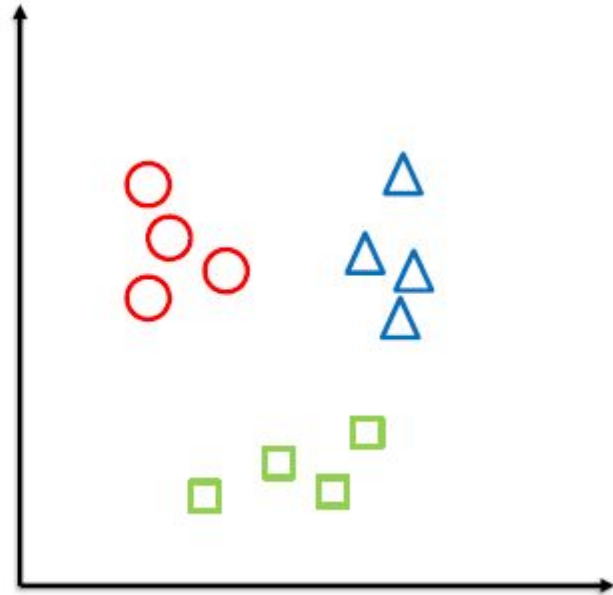


# Multiclass Classification

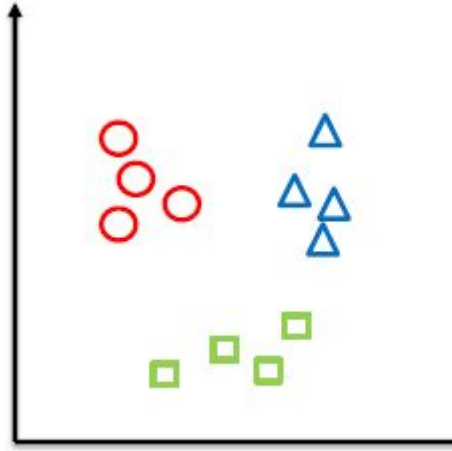
Binary classification



Multiclass classification



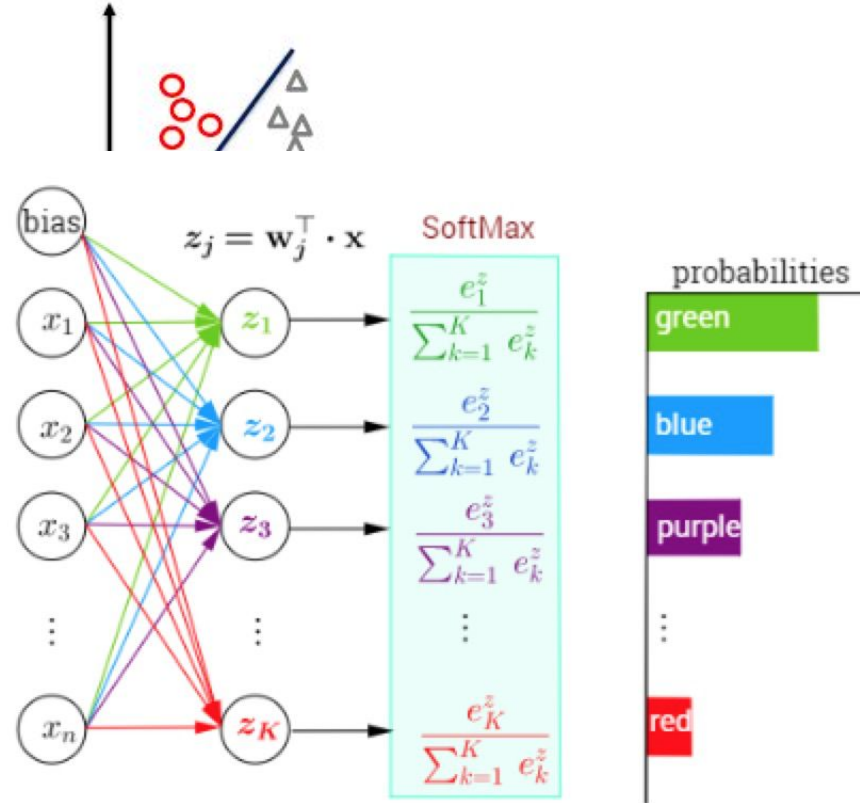
# One-v.s.-All (One-v.s.-Rest)



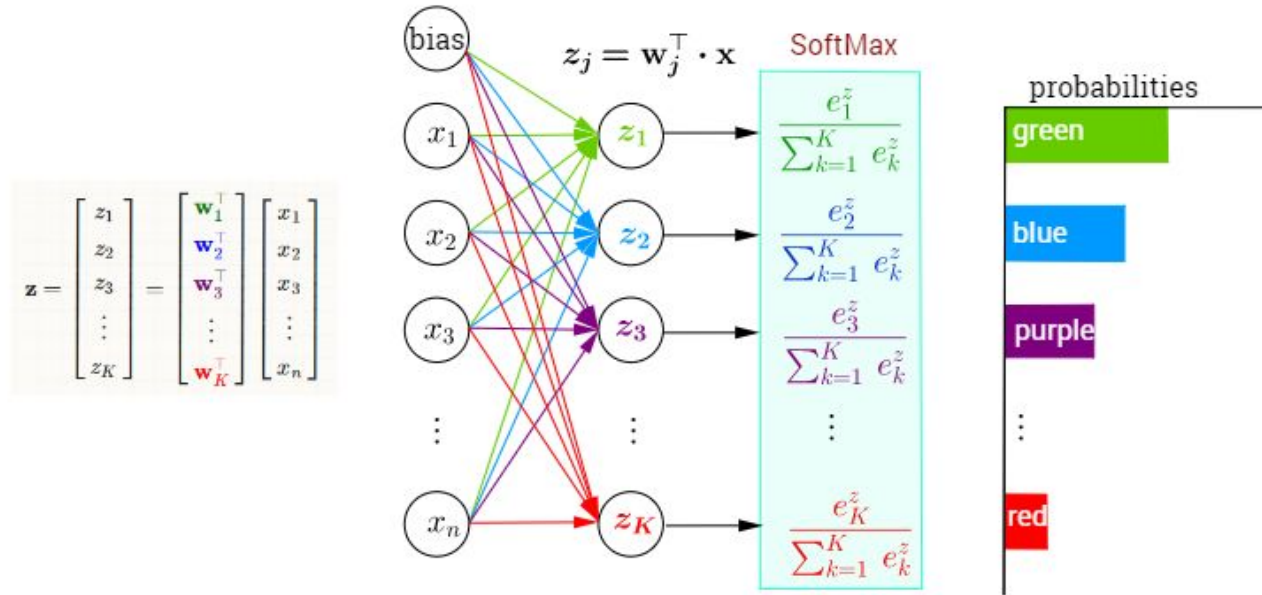
Train a LR classifier  $p_i$   
each class  $i$  to predict the  
of  $y = i$

$$i^* = \arg \max_i p_i$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



# Softmax Classifier



<https://stats.stackexchange.com/questions/265905/derivativ-e-of-softmax-with-respect-to-weights>

# Generative vs Discriminative

# For Classification

- Generative Approaches :
  - Given feature  $X$  and label  $Y$ , a generative model try to find the joint probability:  $P(X, Y)$
  - How the data was generated?
  - From  $P(X, Y) \rightarrow P(Y|X)$ , then categorize
  - Less Direct, More Probabilistic
- Discriminative Approaches :
  - Given feature  $X$  and label  $Y$ , a discriminative model try to find the joint probability:  $P(Y|X)$
  - Distribution-free Approaches
  - Simply categorizes the data
  - More Direct, Less Probabilistic

# Questions

- Generative and Discriminative?
  - Naive Bayes
  - Logistic Regression

# Naive Bayes Model for Text Generation

- For the index of words in range(1, 2, 3, ....T)
  - Random sample the category  $h_i$  from  $p(h)$  or  $h_i$  is fixed
  - Sample the word from the distribution:  $p(d|h_i)$
- However, it does not consider the words' intrinsic dependency
  - E.g., Probability (read the paper) > Probability(read the movie)
  - The words at index T should depend on previous words (T-1, T-2, T-3,...)
- **Hidden Markov Model** *partially* solve the above issue