

Classification

Fundamentals of Artificial Intelligence

Fabien Cromieres

Kyoto University

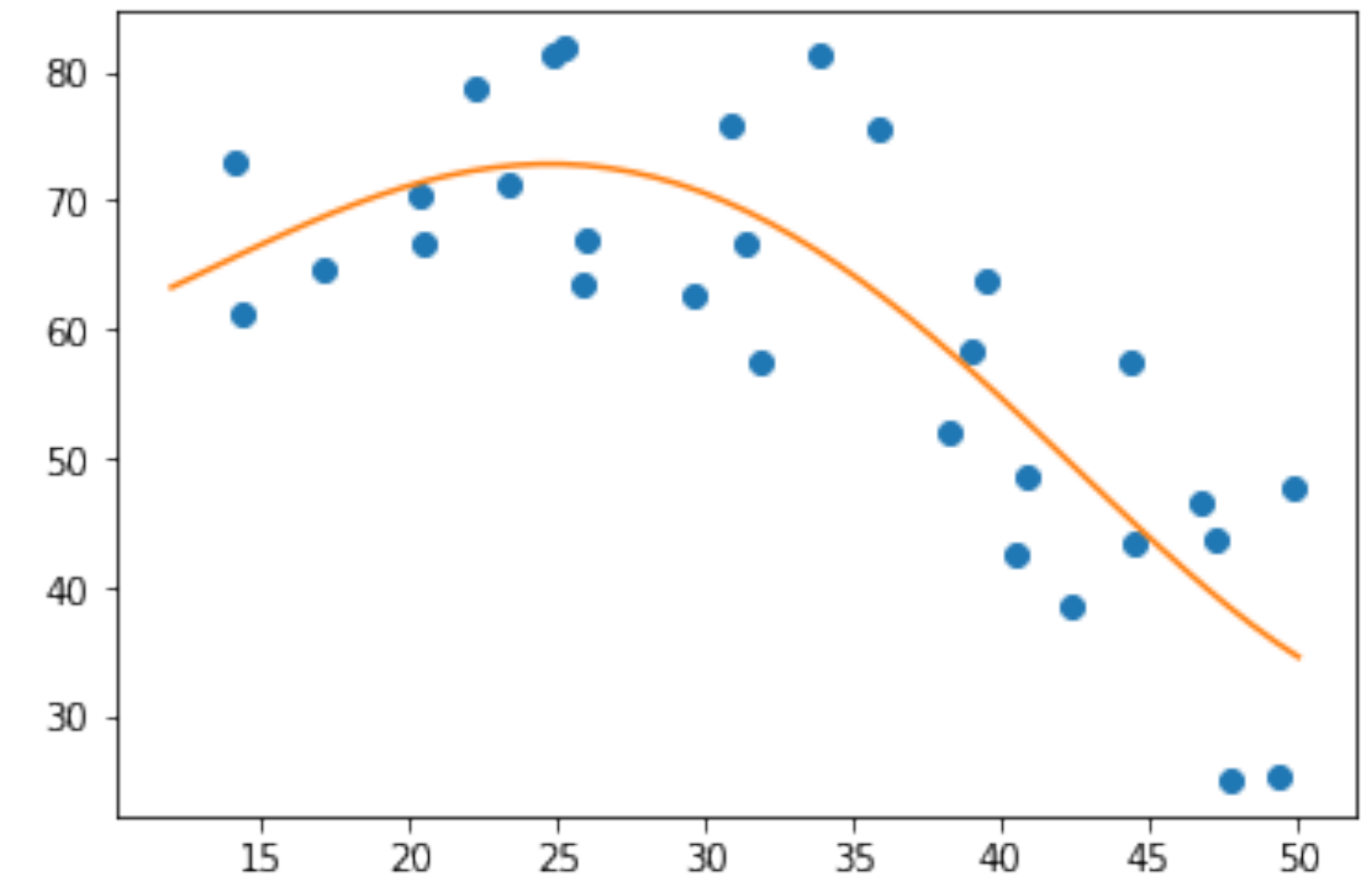
Classification

- In the last two sessions, we considered the task of *regression*
- Given examples, predict a number
 - Age of Death
 - Price of a stock
 - Temperature of an object
 -

Examples

Prediction

Age



BMI

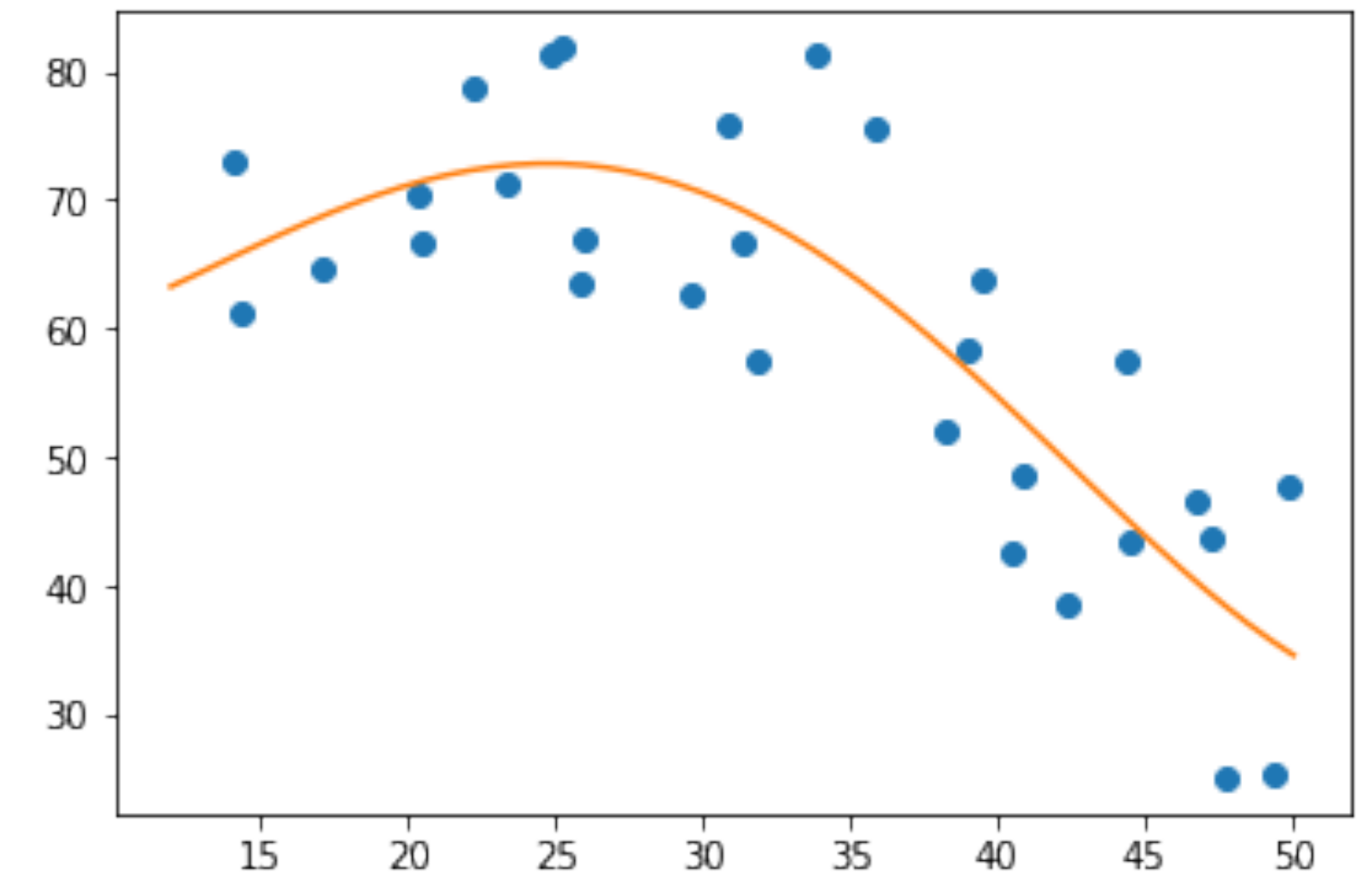
Classification

- In the last two sessions, we considered the task of *regression*
- Given examples, predict a number
 - Age of Death
 - Price of a stock
 - Temperature of an object
 -
- In contrast, today, we are going to consider the task of *classification*
 - Given examples, predict a **class**

Examples

Prediction

Age



BMI

Classification

- The simple type of classification is called ***binary classification***
- **binary** means that we have 2 classes
- A binary classifier can answer questions of the type:
 - Yes or no?
 - Cat or Dog?
 - Red or Blue?
 - Healthy or Unhealthy?

Binary Classification

Inputs:



Binary Classifier:

Cat or Dog?



Predicted Class:

Cat

Dog

Cat

Cat

Binary Classification

Inputs:

Symptoms:
Coughing
Body Temperature = 40
Headache

Symptoms:
Coughing
Body Temperature = 39
Diarrhea
Weight loss

Symptoms:
Body Temperature = 38
Shivering
Nose bleeding

Binary Classifier:

Has Malaria?

Predicted Class:

Yes

Yes

No

Multiclass Classification

- ***Multiclass classification*** is when we have more than 2 possible answers:
 - What animal is that? (cat, dog, lion, bird, fish, human, other) [7 classes]
 - Which disease is that? (Malaria, Flu, Dengue, Tuberculosis, other) [5 classes]
 - Which color is that? (red, blue, white, yellow, green, black) [6 classes]

Multiclass Classification

Inputs:



Multiclass Classifier
(7 classes)

Which animal? (cat, dog, lion, bird, fish, human, other)



Predicted Class:

Cat

Dog

Bird

Cat

Binary Classification

Inputs:

Symptoms:
Coughing
Body Temperature = 40
Headache

Symptoms:
Coughing
Body Temperature = 39
Diarrhea
Weight loss

Symptoms:
Body Temperature = 38
Shivering
Nose bleeding

Multiclass Classifier
(5 classes)

Which disease? (Malaria, Flu, Dengue, Tuberculosis, other)

Predicted Class:

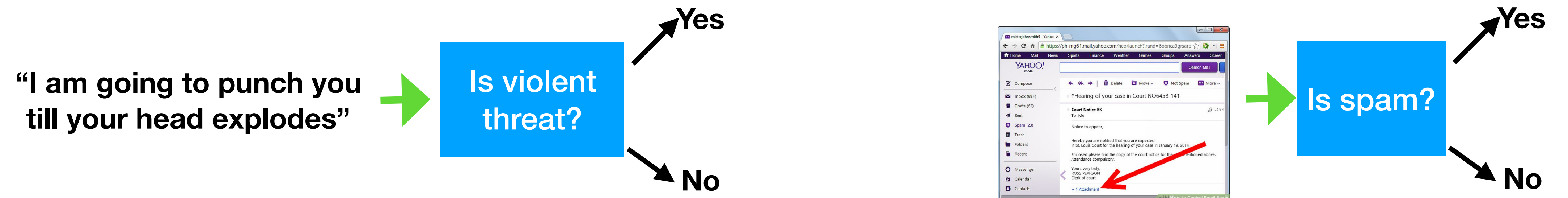
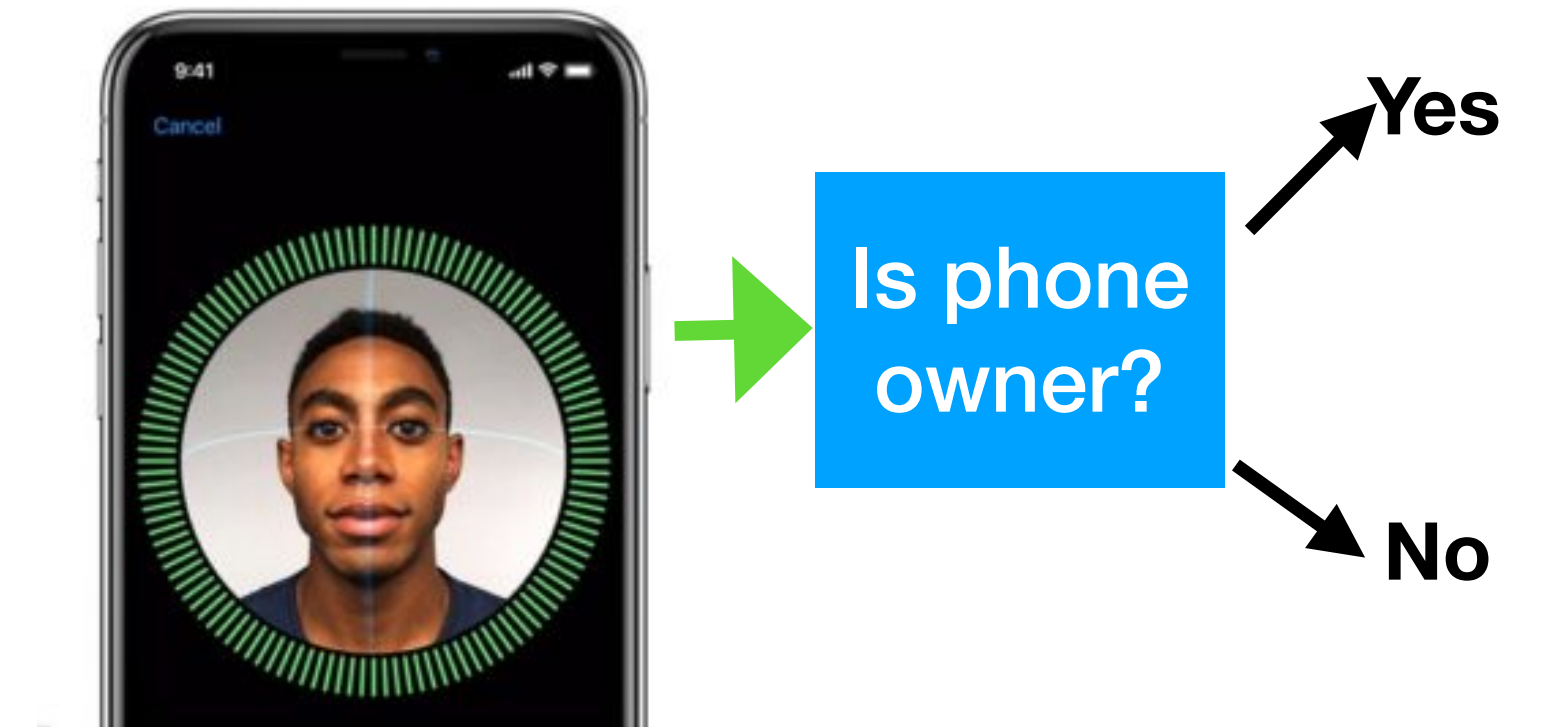
Dengue

Malaria

Other

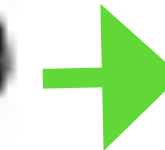
Classification

- Many interesting problems in AI can be seen as Classification problems
- Binary Classification:
 - Biometry-based identification (such as Face-id):
 - Spam detection
 - Automatic detection of use of bullying in social media
 - ...



Classification

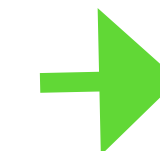
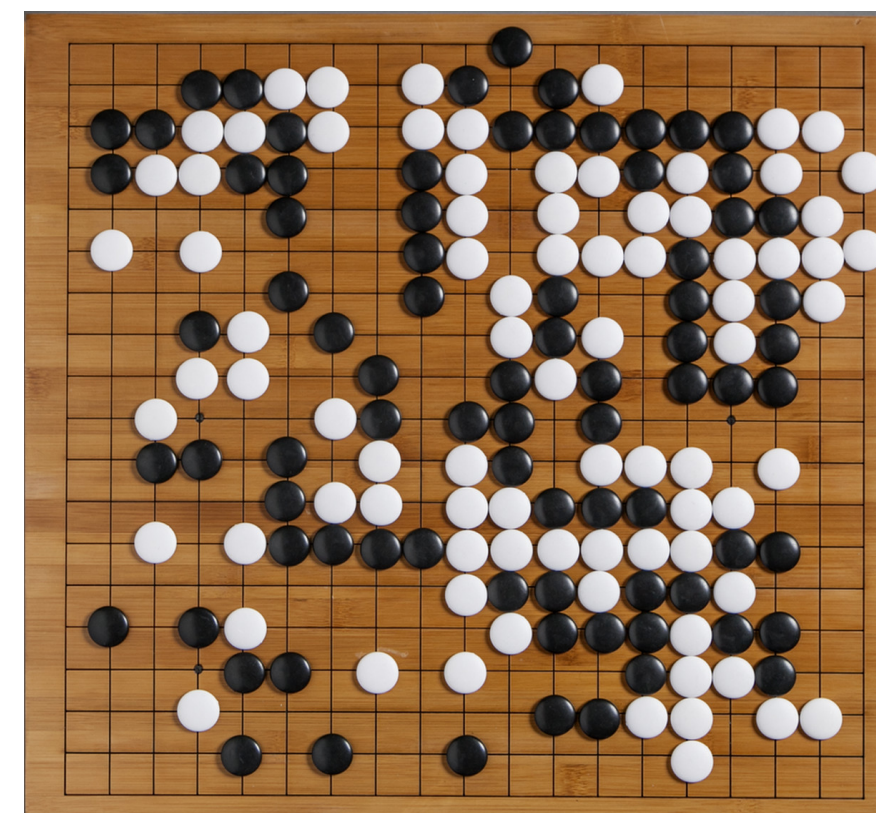
- Some interesting problems that can be seen as **multiclass** classification:
 - Optical Character Recognition (OCR)
 - AI for games
 - Tagging photos with people name
 -



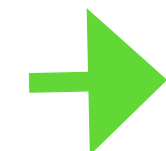
Which letter?
(A-Z)
26 classes



Q



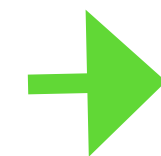
Where to play my
next stone?
19x19 classes



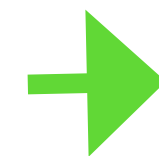
(10,12)

Multiclass vs Multilabel

- In multiclass classification, classes are **exclusives**
 - For each input there is one and only one correct class
- If we want that an input may have more than one class, this is called ***multilabel classification***
- But we will **not** consider this case for now



Which facial
features?



Mustache

Beard

Short hair

Binary Classification: Predicting vote

- As a practical example of **binary** classification, let us try to predict the **vote** of somebody given some features
 - For now, we only consider income as a feature
 - (consider it expressed in 1000\$/year or 10万円/年 to get a concrete idea)
 - We only consider 2 parties for now: Left-wing party and Right-wing party
- We did a survey of 30 persons that accepted to answer us

	income	vote
0	39.0	L
1	30.0	L
2	47.0	L
3	69.0	R
4	52.0	R
5	110.0	R
6	...	

- Now, given the income of somebody else, we want to **predict** for which party he is going to vote
- Useful task to:
 - Predict the result of an election (if we know the income of each person in the country, we can predict their vote efficiently)
 - Do efficient marketing on an individual

Binary Classification: Predicting vote

- Numbers are easier to use than letters in Machine Learning
- As a first step, we assign number 0 to class “*Left-Wing Party*” and number 1 to class “*Right-Wing Party*”
 - Which number you assign to which category will have no influence on prediction

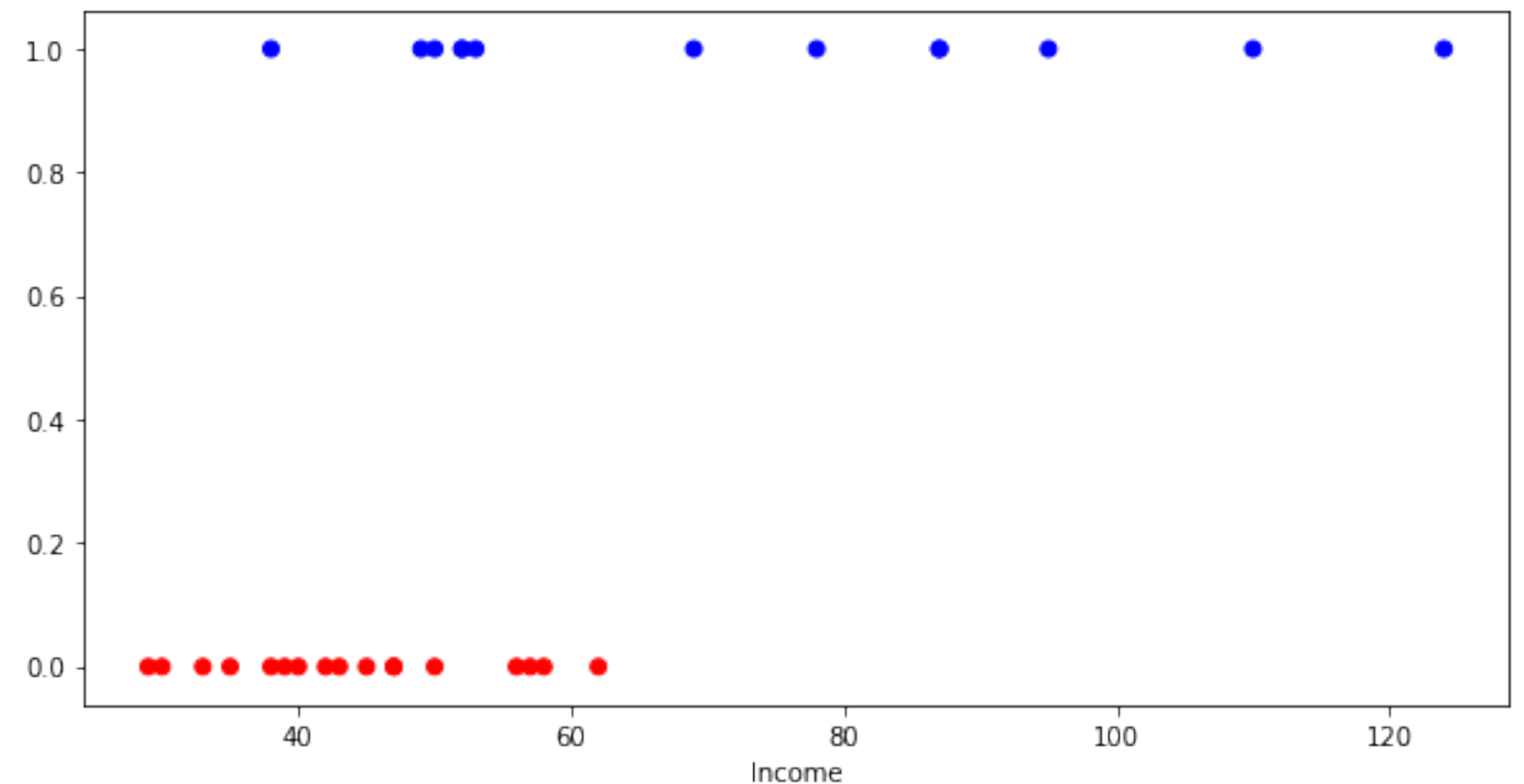
	income	vote
0	39.0	L
1	30.0	L
2	47.0	L
3	69.0	R
4	52.0	R
5	110.0	R
6	...	

	income	vote
0	39.0	0
1	30.0	0
2	47.0	0
3	69.0	1
4	52.0	1
5	110.0	1
6	...	

Binary Classification: Predicting vote

- Now, our 30 persons can be visualized like this:
 - Red: Left-wing party
 - Blue: Right-wing party

	income	vote
0	39.0	0
1	30.0	0
2	47.0	0
3	69.0	1
4	52.0	1
5	110.0	1
6	...	



Binary Classification: Predicting vote

- How to predict a class?
- We are going to give a **score** to the possibility that a given voter is of **class 1** (ie. vote for the *right-wing party*)
- Because it is a **binary classification**, we do not need to compute a score for the other class

	income	vote
0	39.0	0
1	30.0	0
2	47.0	0
3	69.0	1
4	52.0	1
5	110.0	1
6	...	

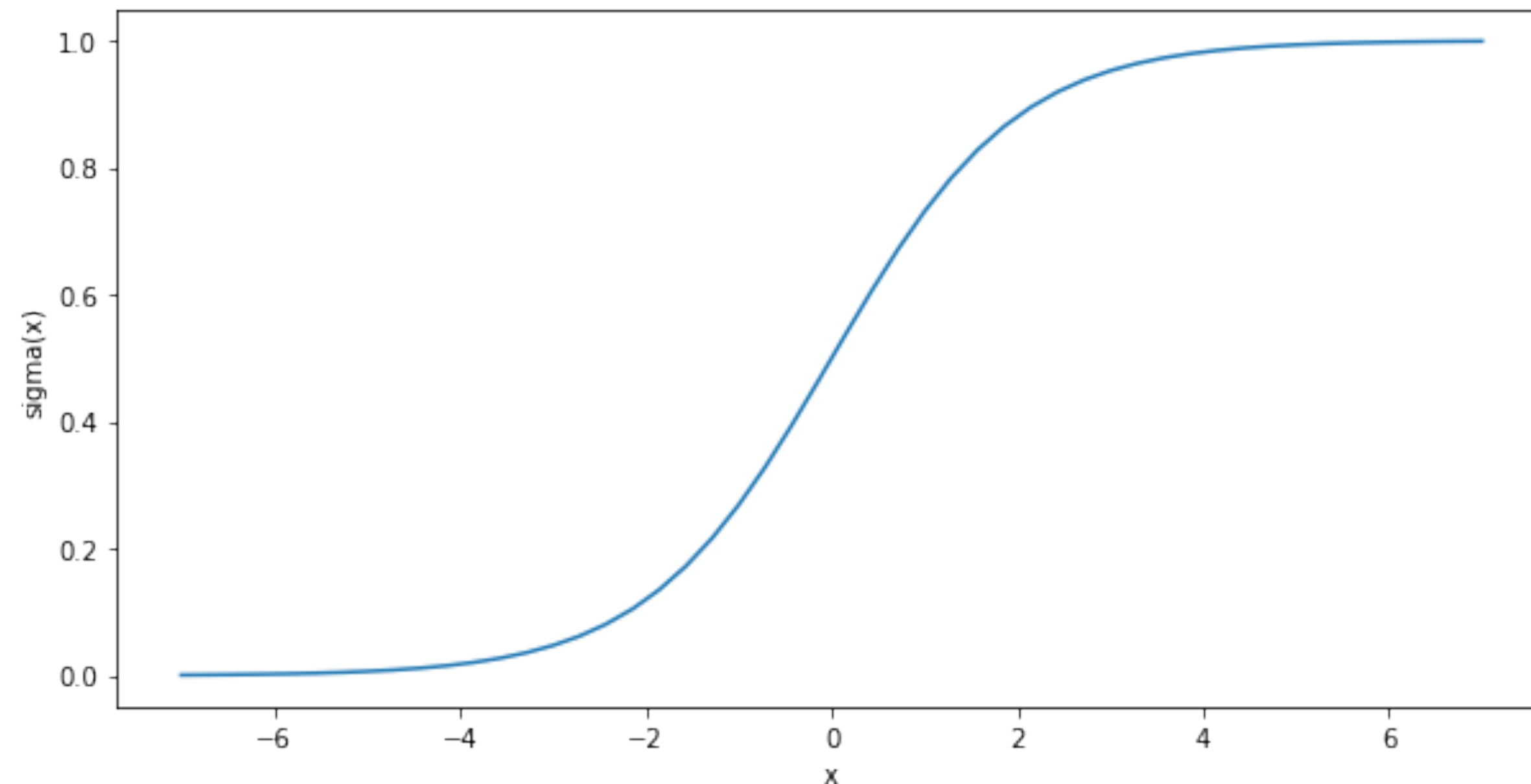
- The score could be any function of the features
- But this time, we will use a linear function again:

$$score(income) = \theta_0 + \theta_1 \times income$$

The Logistic function

- We will then apply the ***logistic function*** to the score
- The logistic function (also called ***sigma function***) is defined by:

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



- If *score* is very large, $\sigma(\text{score})$ will be close to 1
- If *score* = 0 then $\sigma(\text{score}) = 0.5$
- If *score* is very negative, $\sigma(\text{score})$ will be close to 0

Binary Classification: Predicting vote

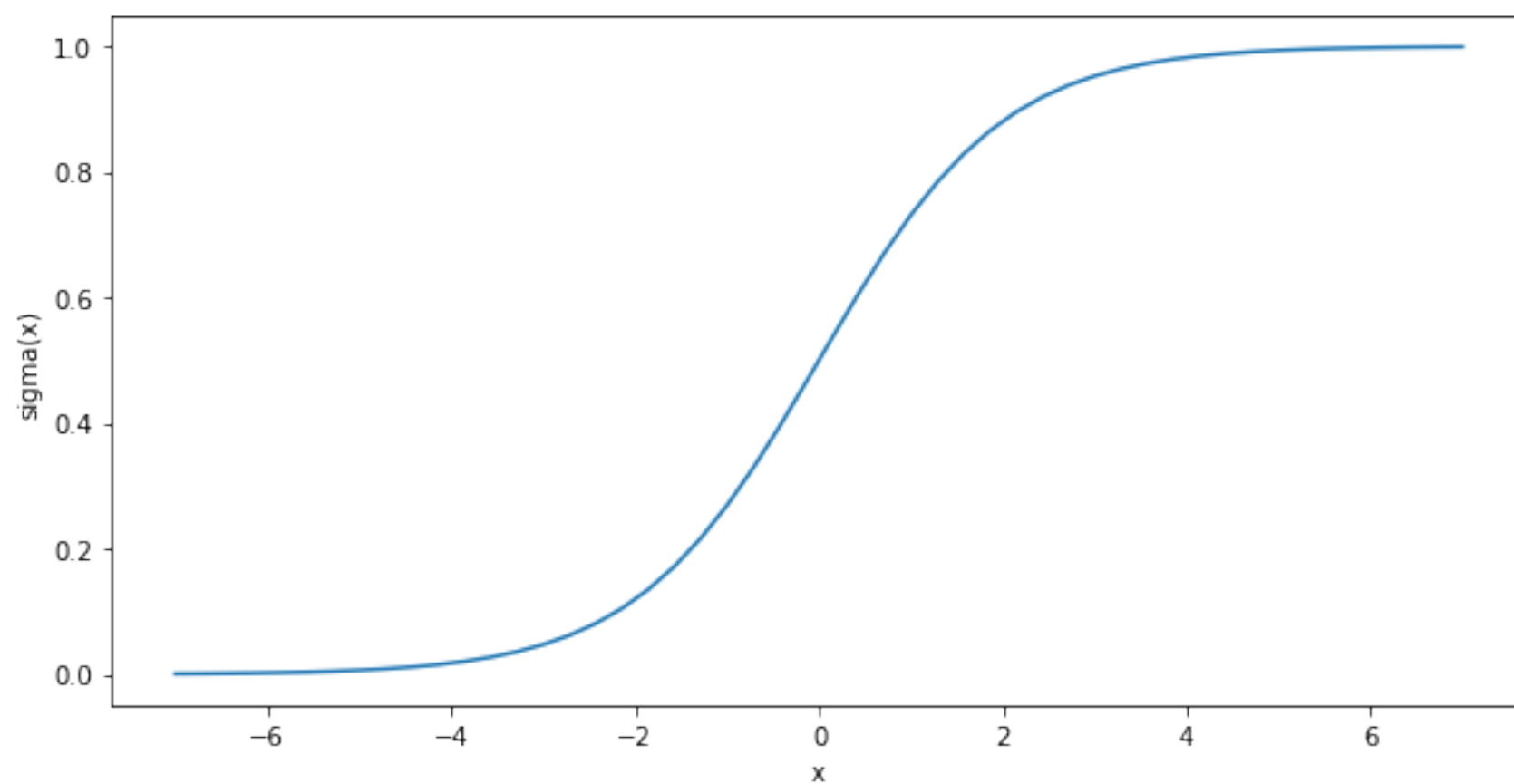
	income	vote
0	39.0	0
1	30.0	0
2	47.0	0
3	69.0	1
4	52.0	1
5	110.0	1
6	...	

- Therefore, our classification model is like this:

$$score(income) = \theta_0 + \theta_1 \times income$$

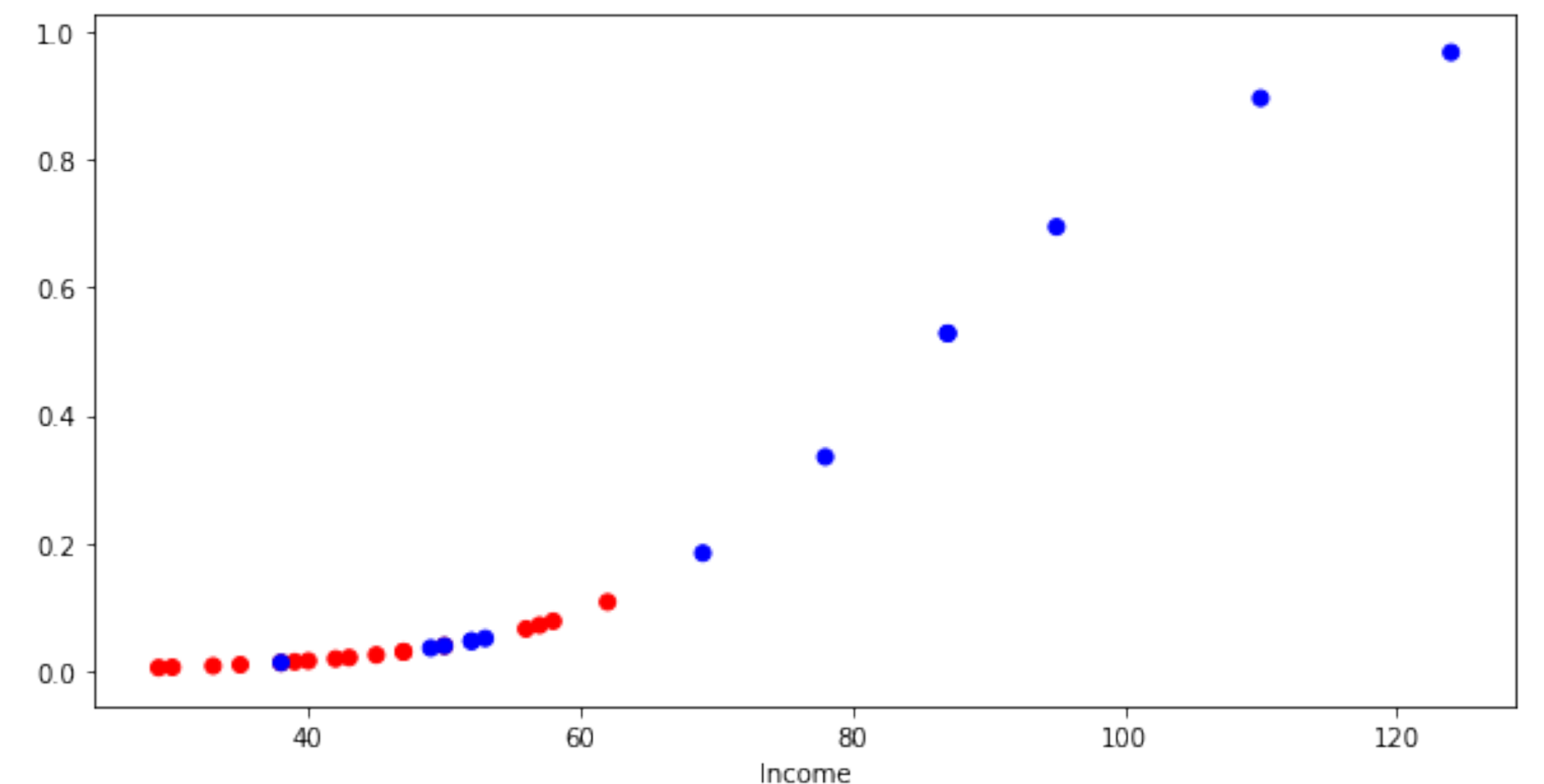
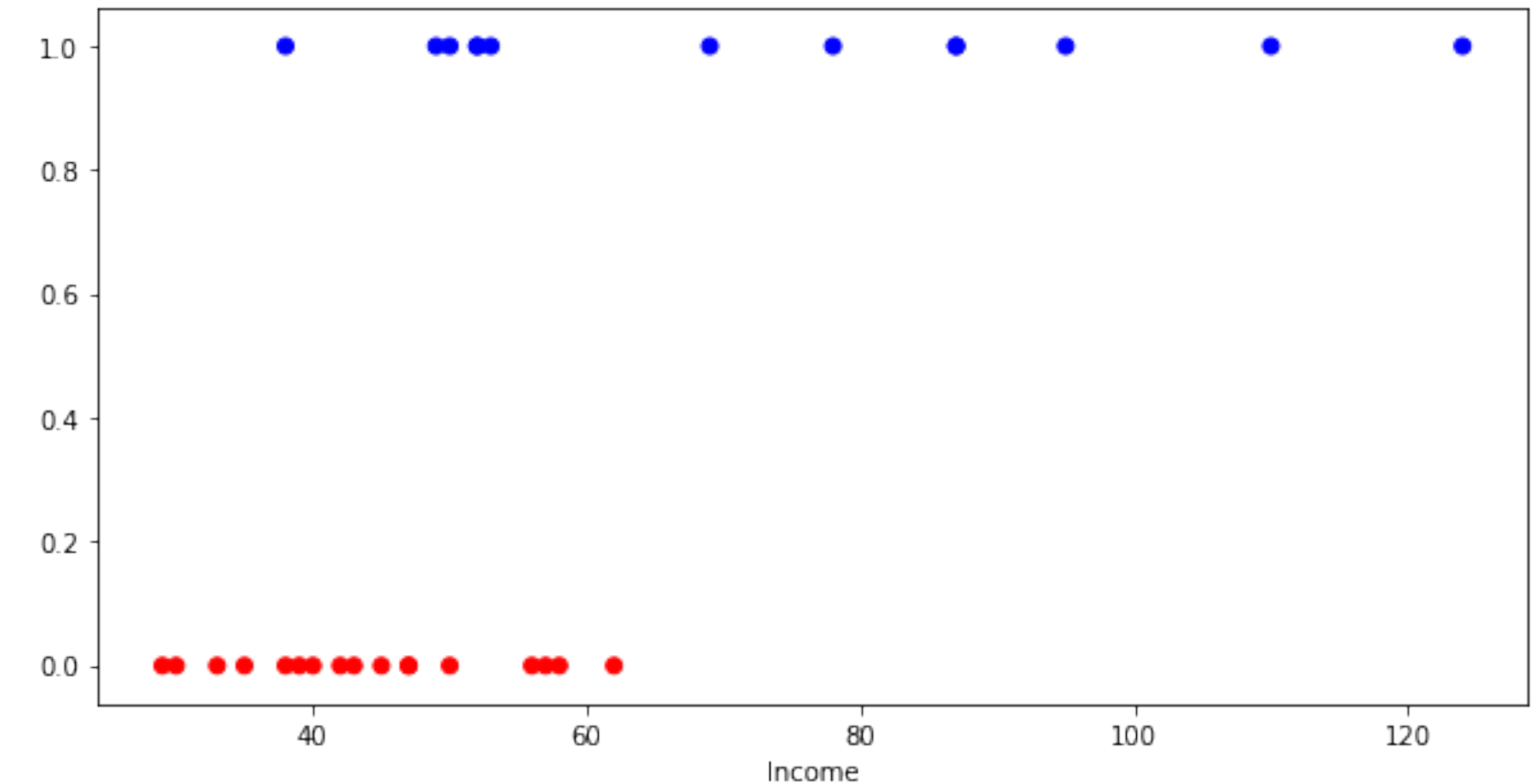
$$V_{model} = \sigma(score)$$

- In practice, the predicted **V** will not be equal to **0** or **1**
- The predicted **V** is a value between **0** and **1**
- We interpret **V** as the probability that a given voter will vote for the right wing party
- We can also interpret **V** as the confidence of the model that a given voter will vote for the right wing party



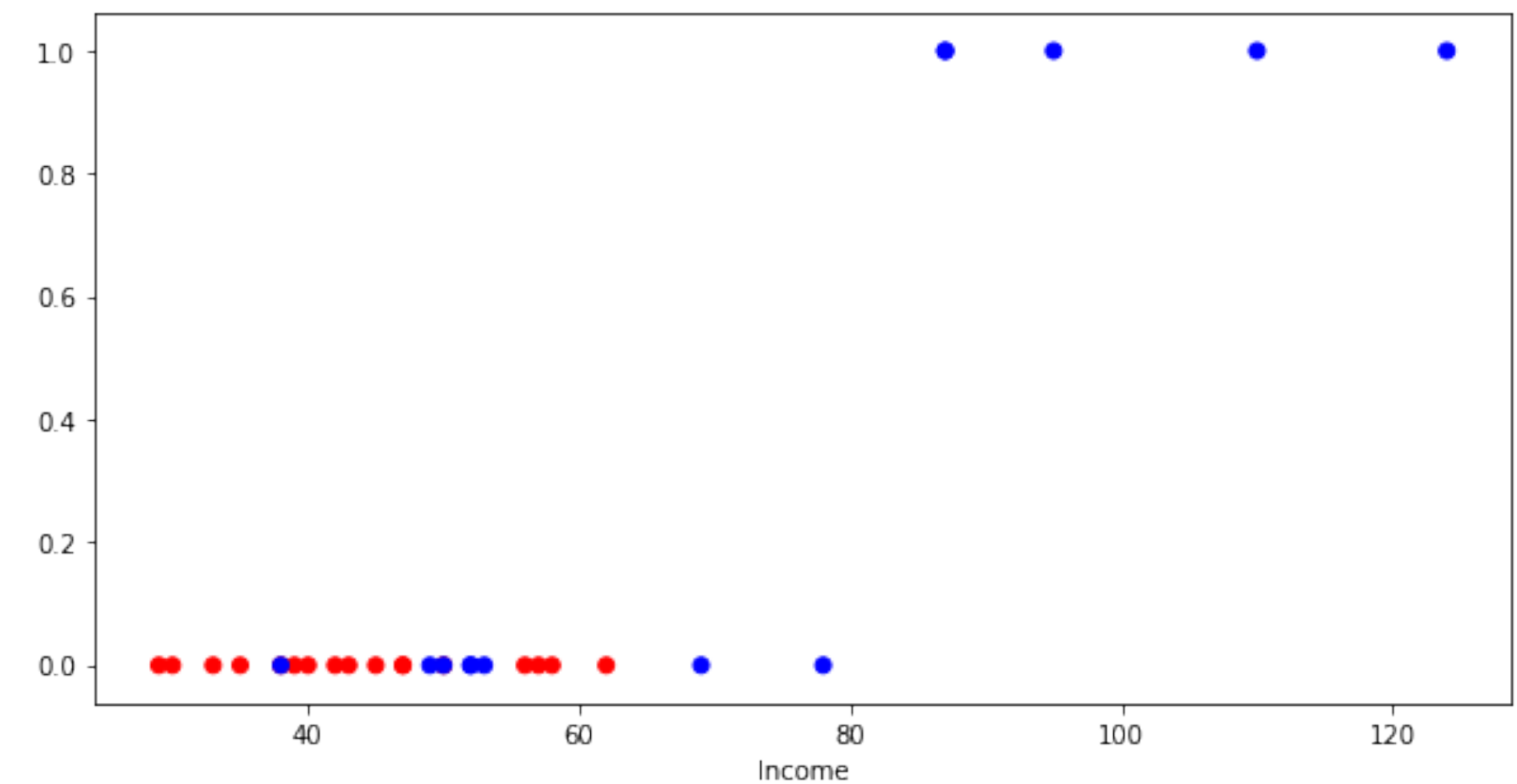
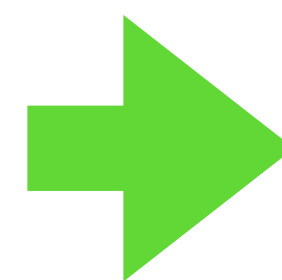
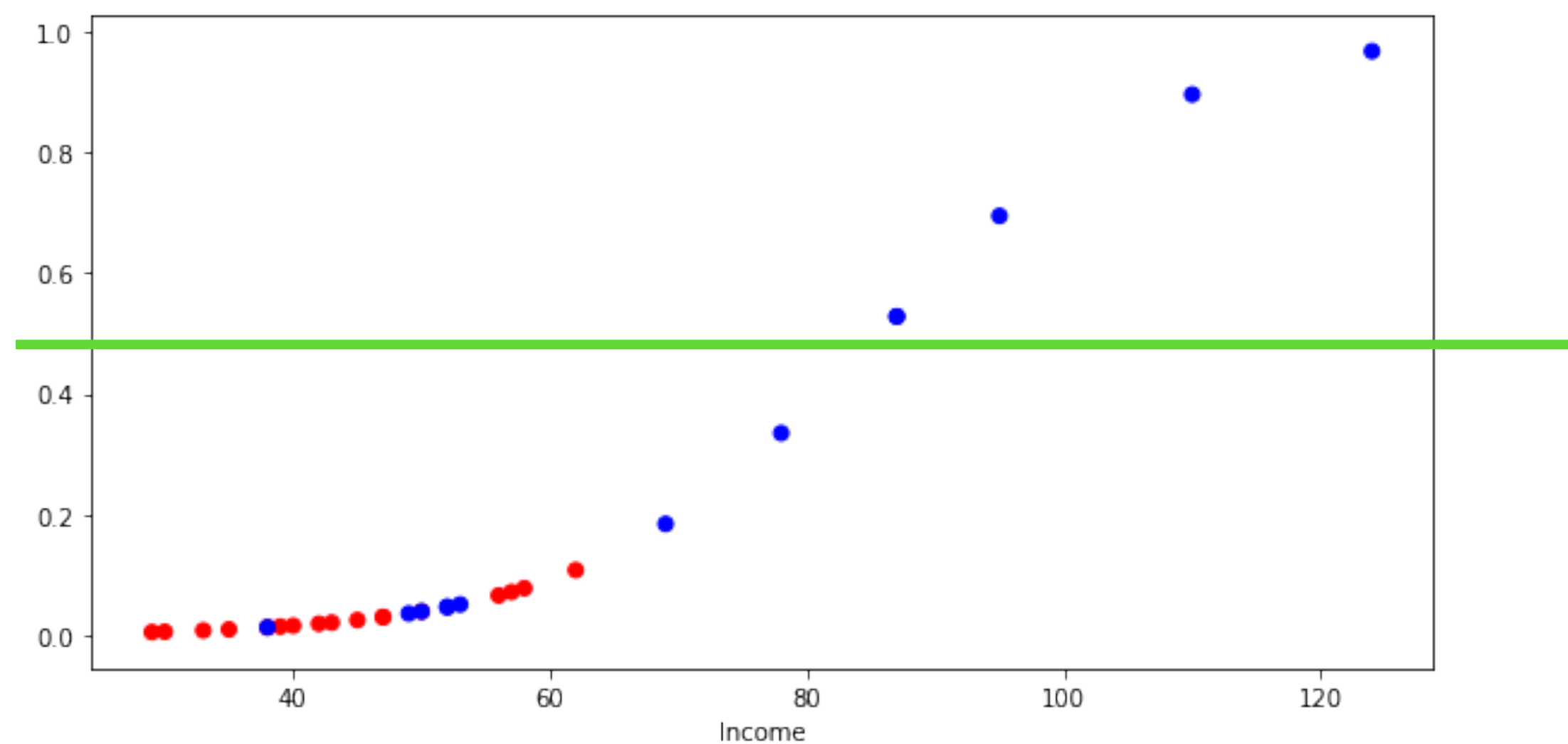
Difference between prediction and data

- Our examples represent binary values
- Either **0** or **1**
- Our predictions are probabilities between **0** or **1**
- We want to predict the data: either **0** or **1**



Difference between prediction and data

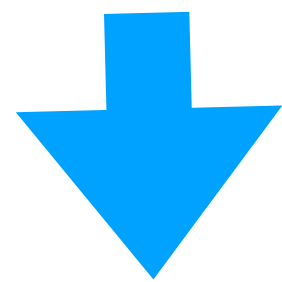
- Our predictions are probabilities between 0 or 1
- We want to predict the data: either 0 or 1
- -> We choose a decision boundary
 - (usually we check if predicted value is higher or lower than 0.5)



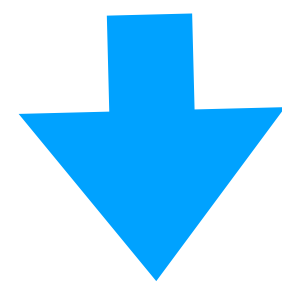
Our model predictions

- Finally, we now have predictions from our model

$$\text{score}(\text{income}) = \theta_0 + \theta_1 \times \text{income}$$



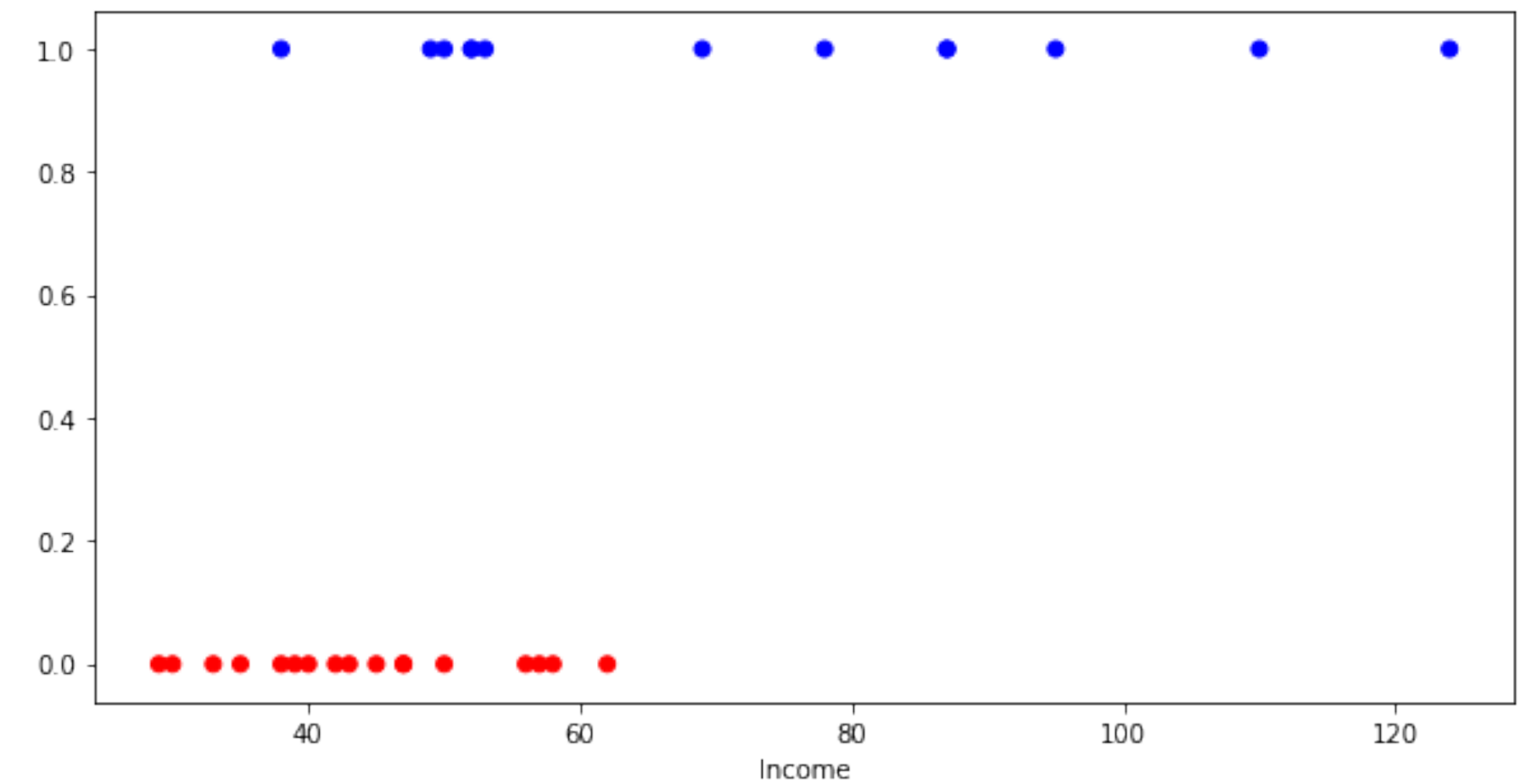
$$V_{\text{model}} = \sigma(\text{score})$$



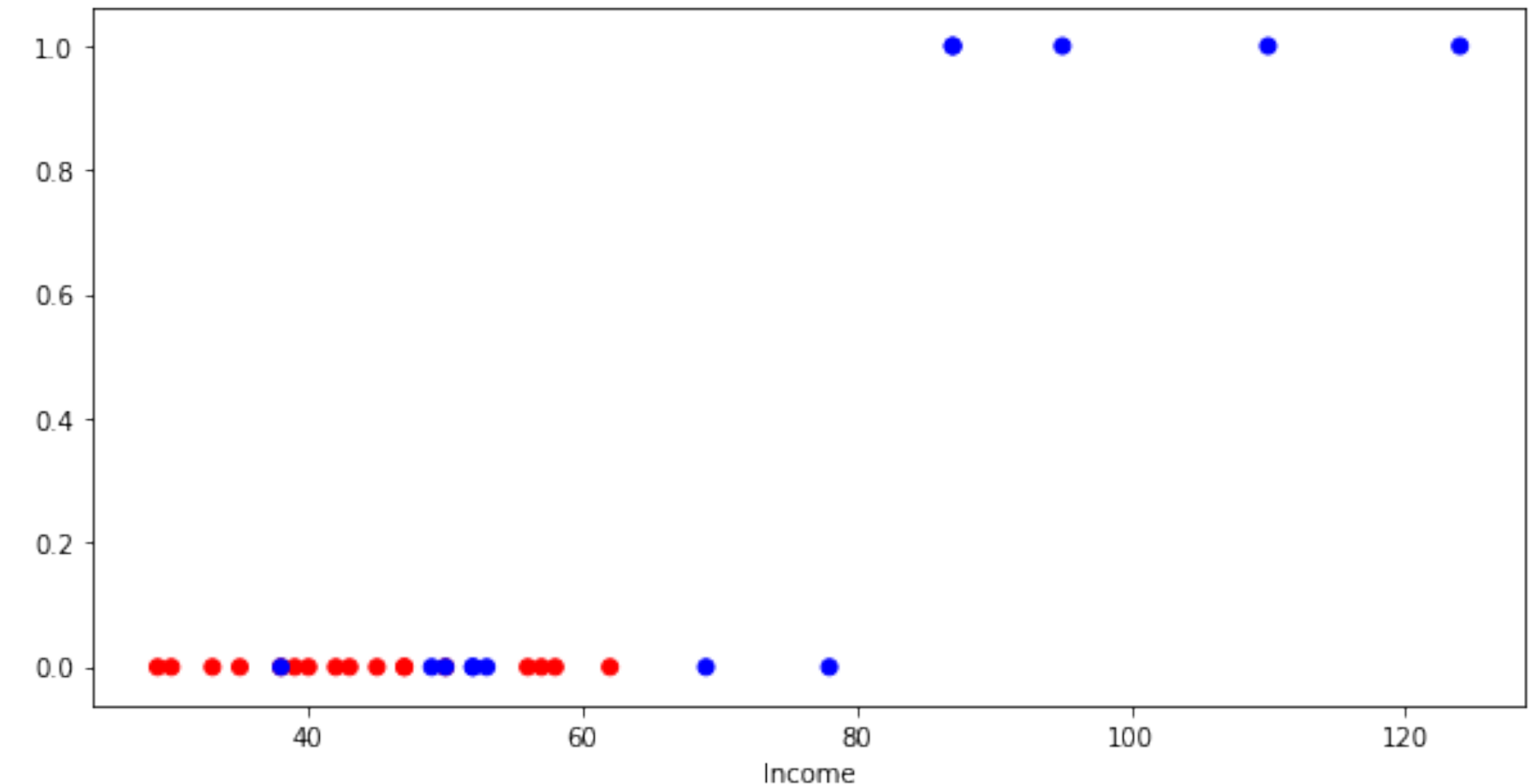
Class = 1 if $V > 0.5$ else 0

- Remaining question: **how to find the parameters θ ?**

Example data

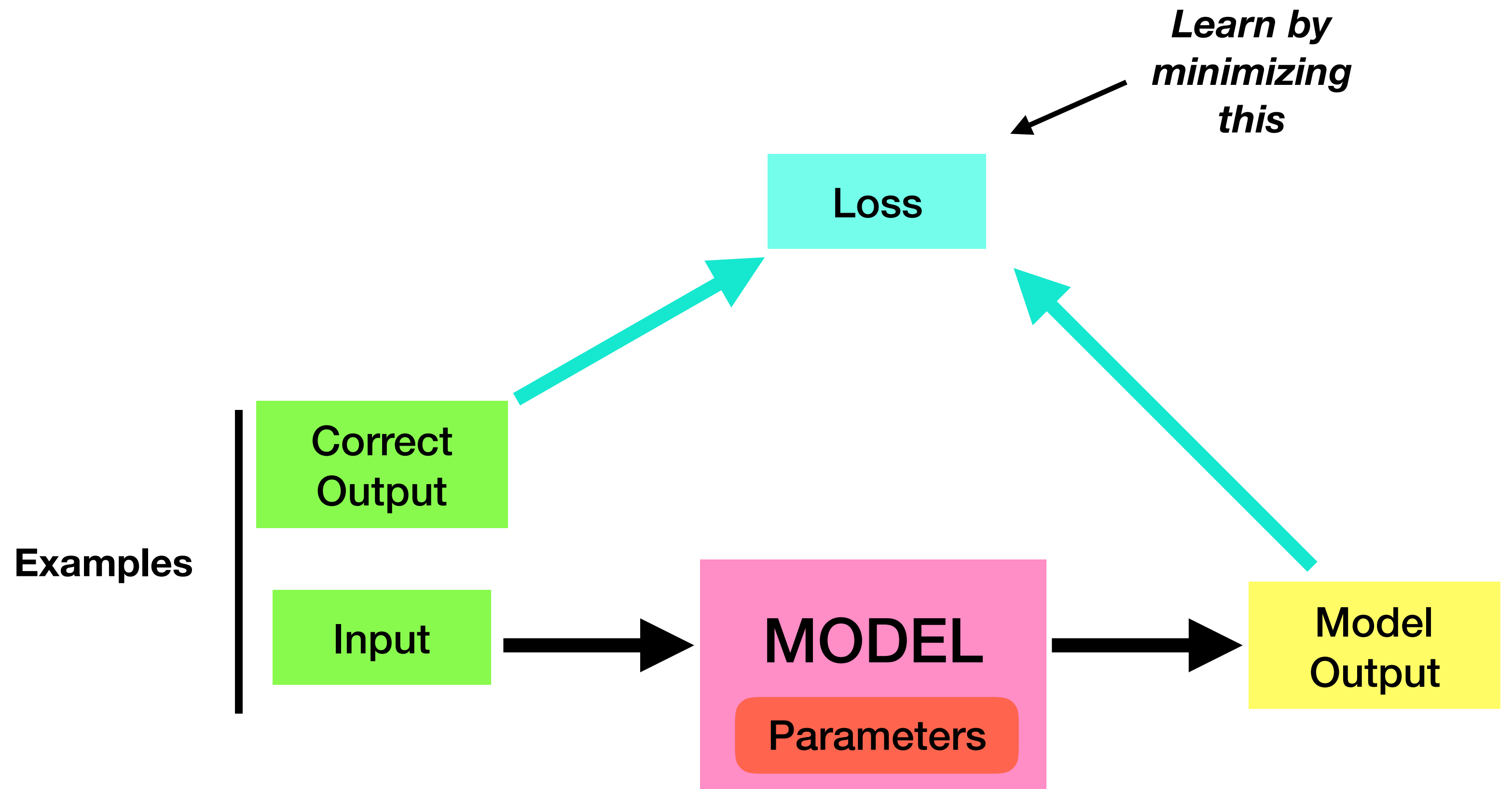


Our model predictions



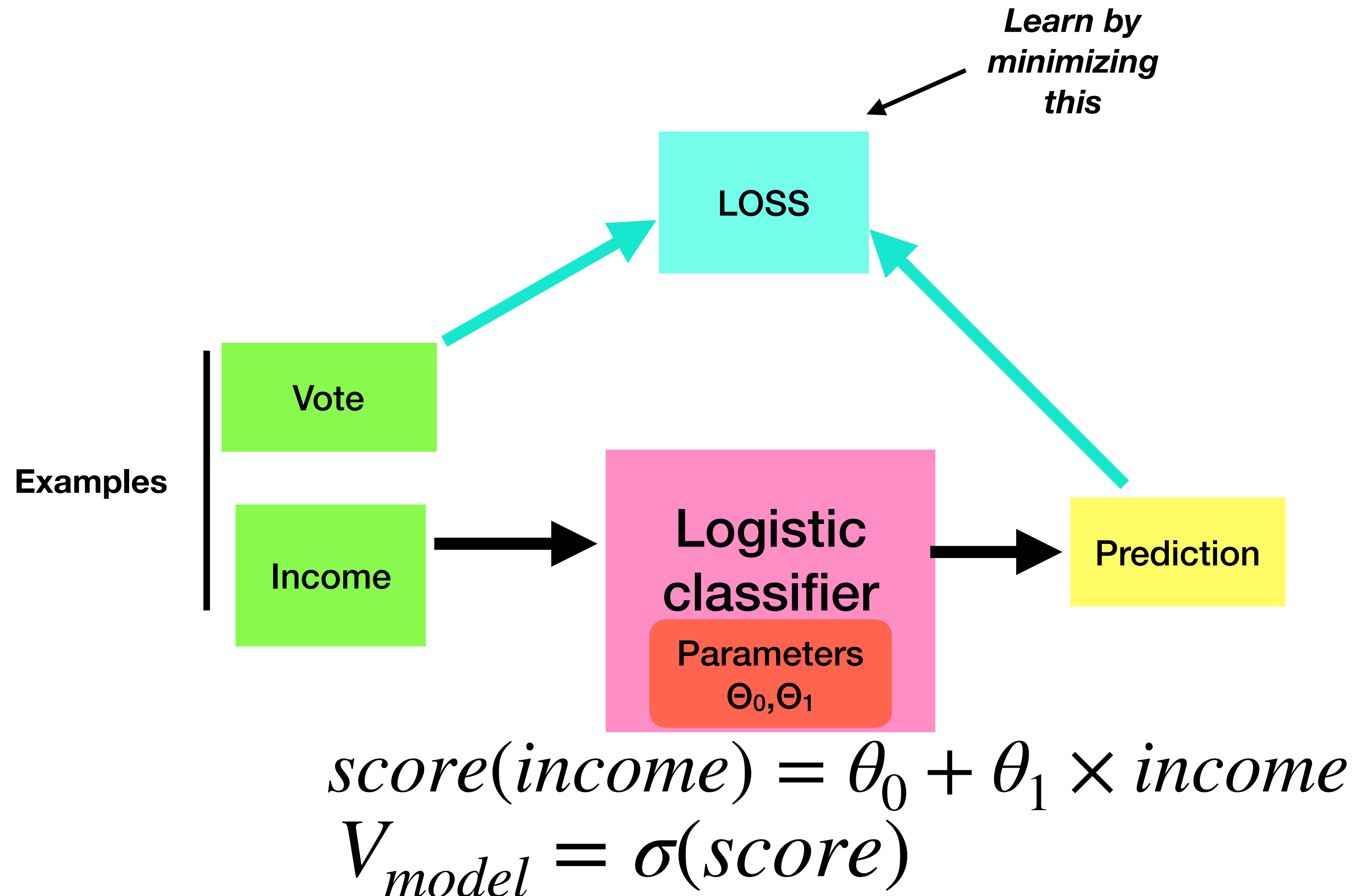
Supervised Learning

- In supervised learning, we usually have:
 - A **MODEL**: a “parameterized” function that takes input and produce output
 - A **Loss**: A function that compute how different the model output is from the correct output
 - **Examples** of input and correct output



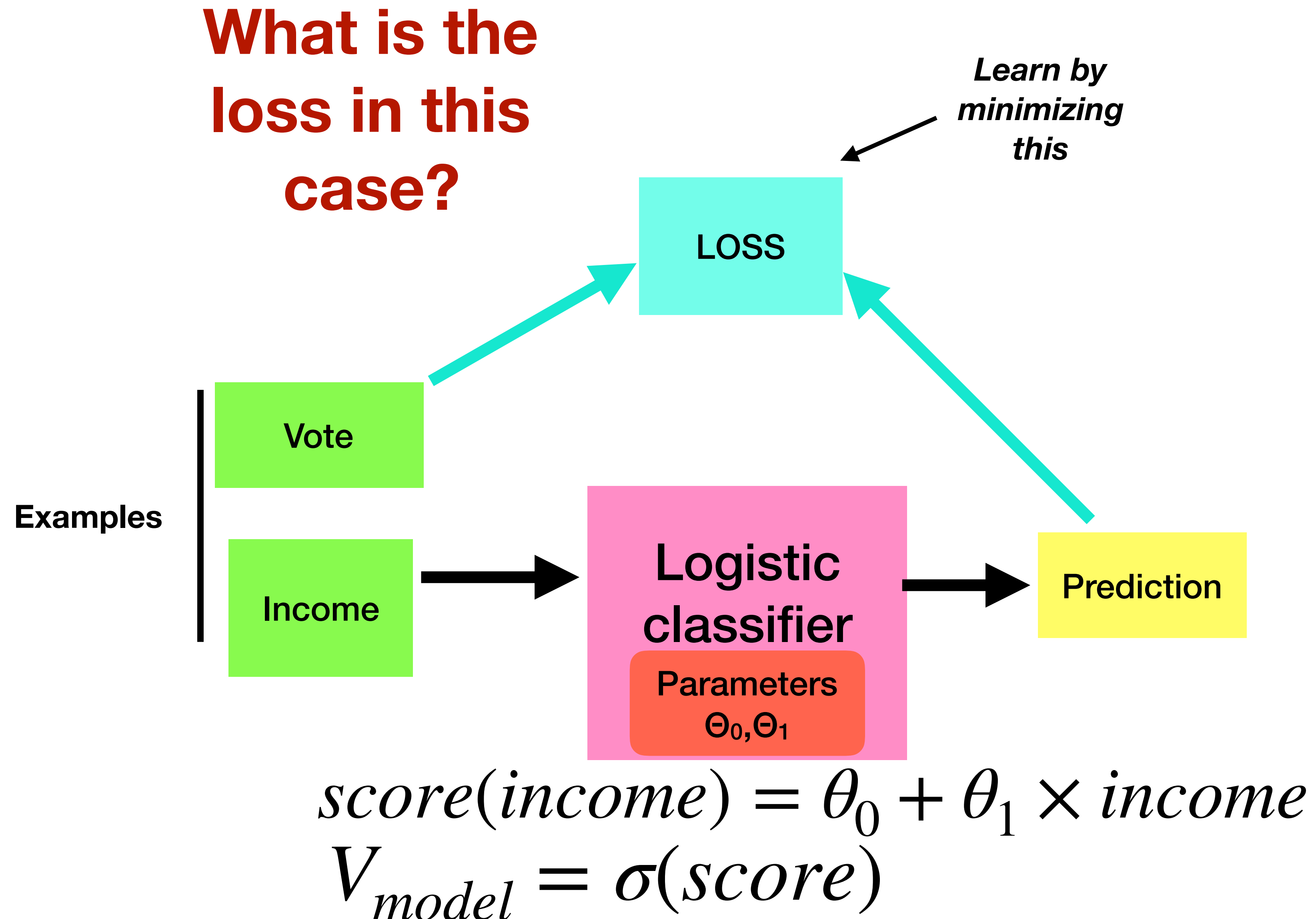
Supervised Learning

- In supervised learning, we usually have:
 - A **MODEL**: a “parameterized” function that takes input and produce output
 - A **Loss**: A function that compute how different the model output is from the correct output
 - **Examples** of input and correct output (cigaretts smoked, age of death)



Supervised Learning

- In supervised learning, we usually have:
 - A **MODEL**: a “parameterized” function that takes input and produce output
 - A **Loss**: A function that compute how different the model output is from the correct output
 - **Examples** of input and correct output (cigarets smoked, age of death)



What is the loss we should use?

- We need a loss that tells us how bad are the predictions given the examples
- One possibility: we count how many predictions were wrong

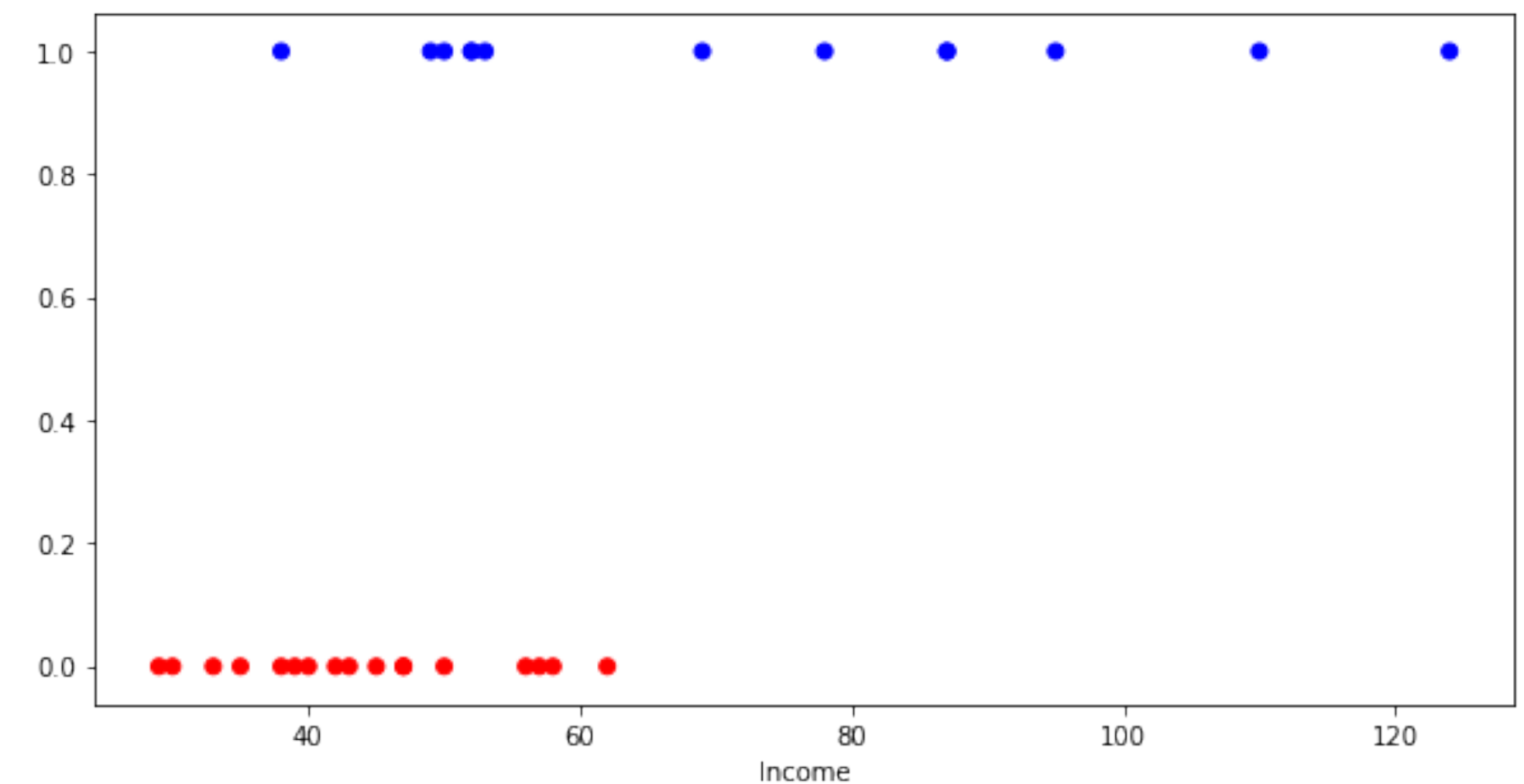
7 right-wing voters were predicted to vote left
0 left-wing voters were predicted to vote right
23 out of 30 predictions are correct

Accuracy: 23/30

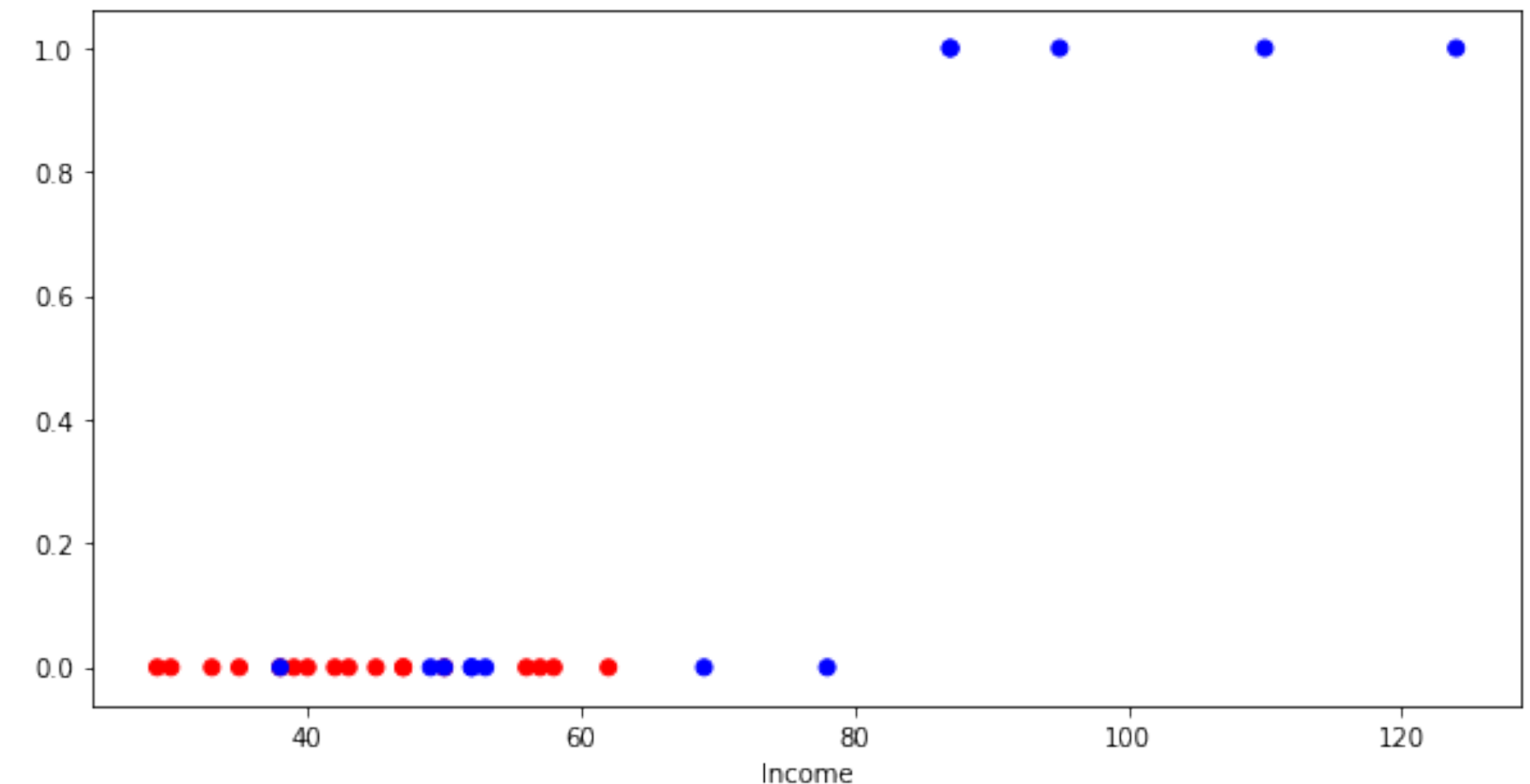
Loss: 7/30

- Intuitively, it seems a good way to evaluate the model
- Unfortunately, it is not a good loss for learning parameters

Example data



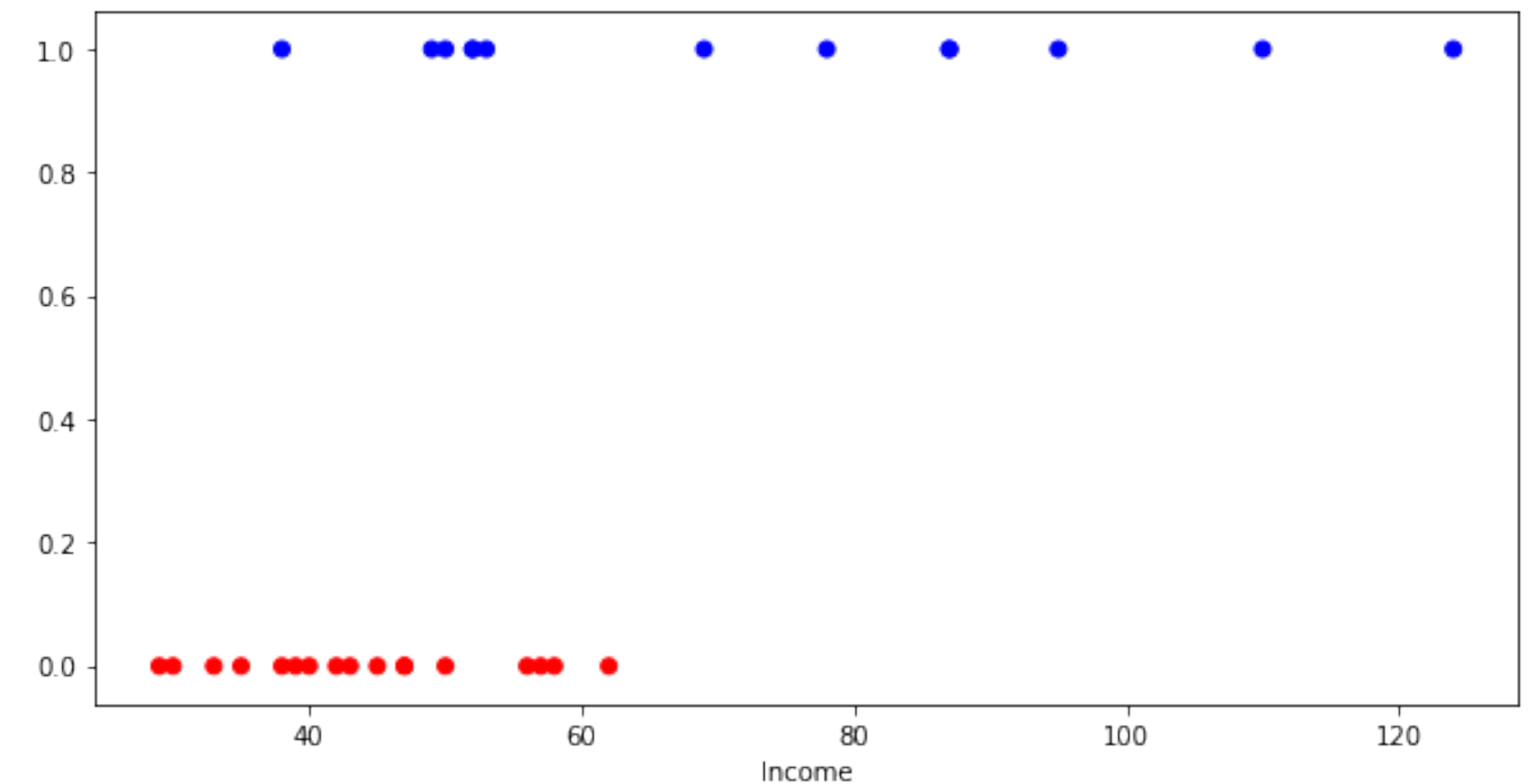
Our model predictions



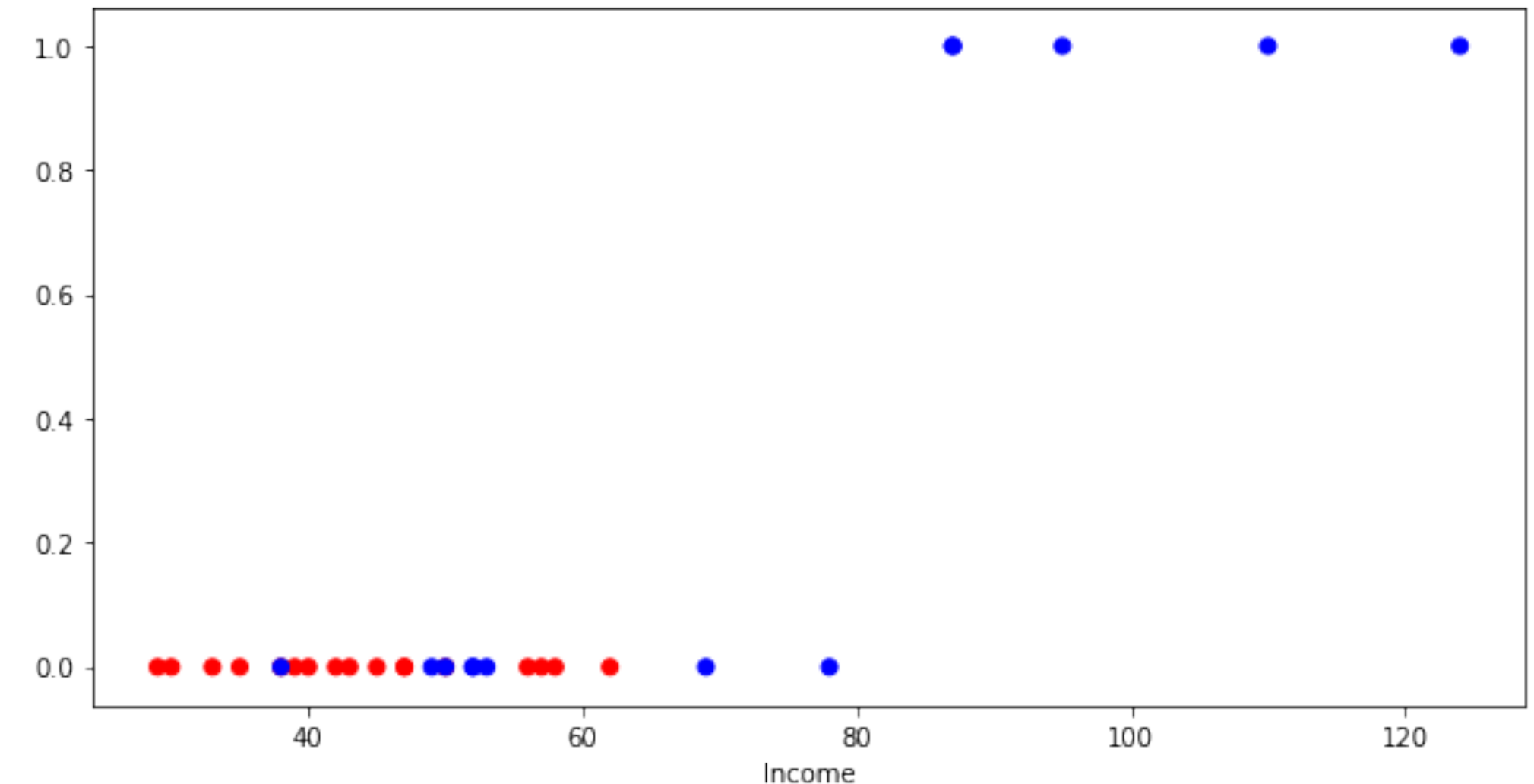
What is the loss we should use?

- We need a loss that tells us how bad are the predictions given the examples
- One possibility: we count how many predictions were wrong

Example data



Our model predictions



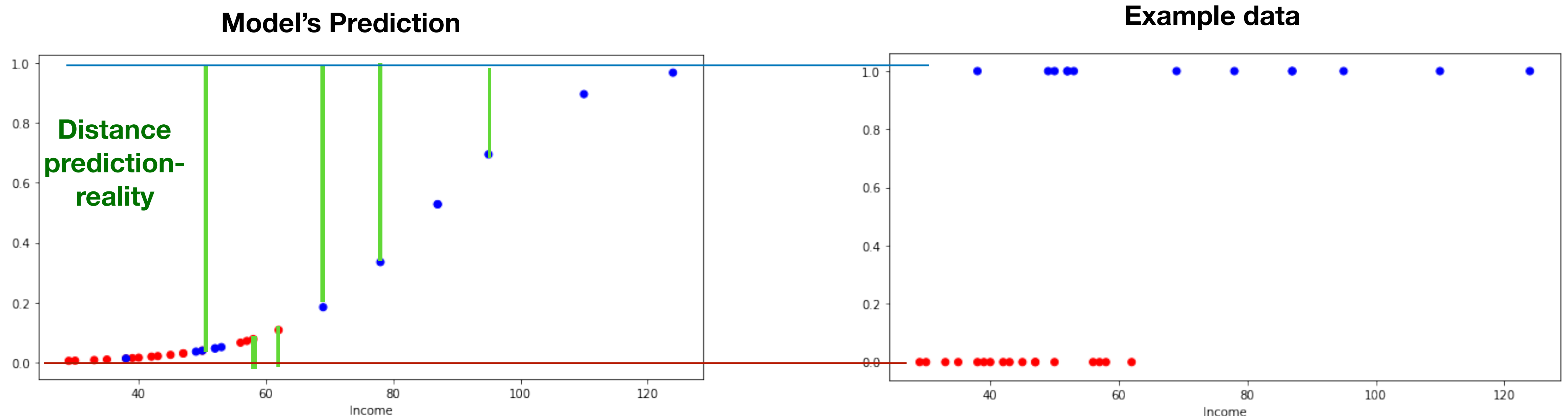
- Intuitively, it seems a good way to evaluate the model
- Unfortunately, it is **not a good loss** for learning parameters
- Because this loss is not a continuous function of the parameters θ
- We cannot compute a gradient -> we cannot use gradient descent

What is the loss we should use?

- How about the *Mean Squared Error*?

$$V_{model} = \sigma(score)$$

- If we use the predicted probability instead of the predicted class
- Then the same situation as with regression: we compute the distance between the prediction and the real value
- In theory it works; in practice, it does not work
- The Mean Squared Error loss of a logistic classifier is very difficult to minimize in practice (for technical reasons we will not discuss here)

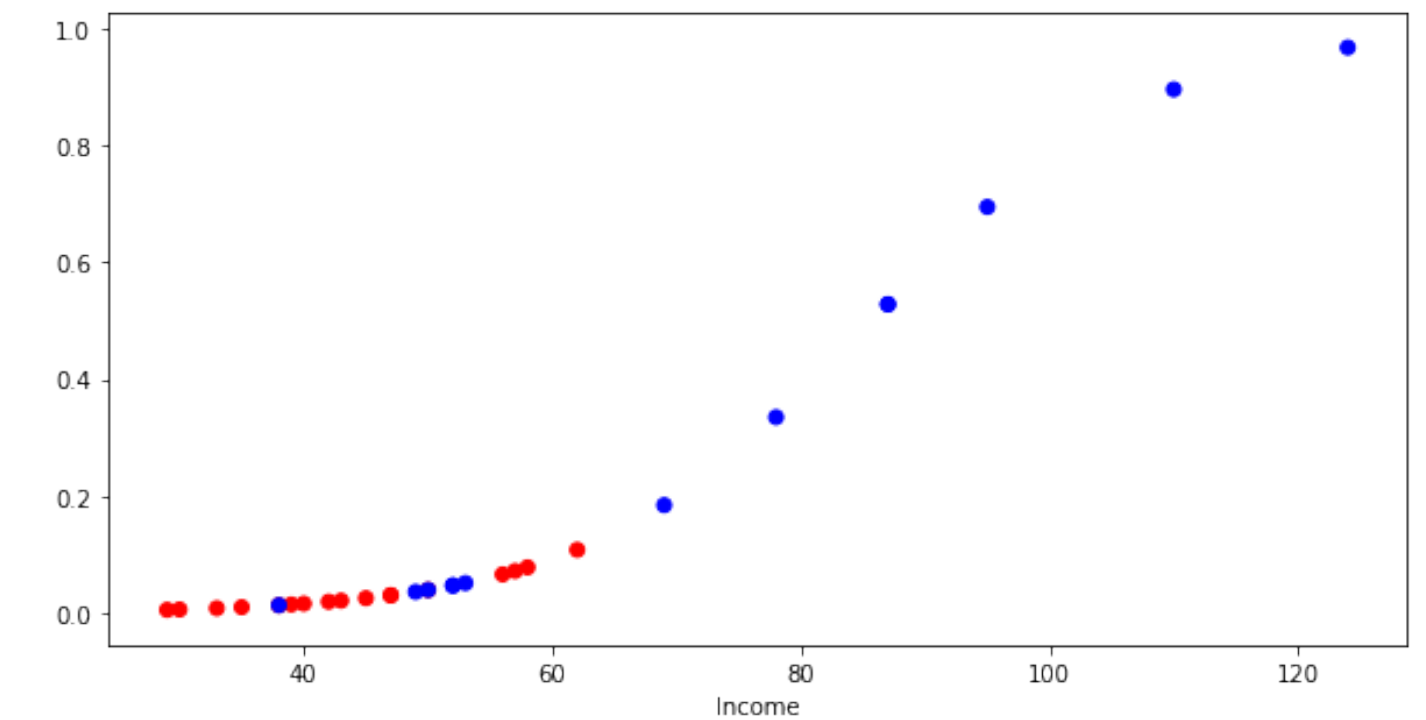


What is the loss we should use?

- We will use something called the ***cross-entropy loss***
- It is defined as follow
- For each example:
 - If the correct class is **1**, then the ***cost*** of the prediction ***V*** is $-\log(\mathbf{V})$
 - If the correct class is **0**, then the ***cost*** of the prediction ***V*** is $-\log(1-\mathbf{V})$
- Then we average the cost over each example

$$CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N cost_i$$

$$V_{model} = \sigma(score)$$



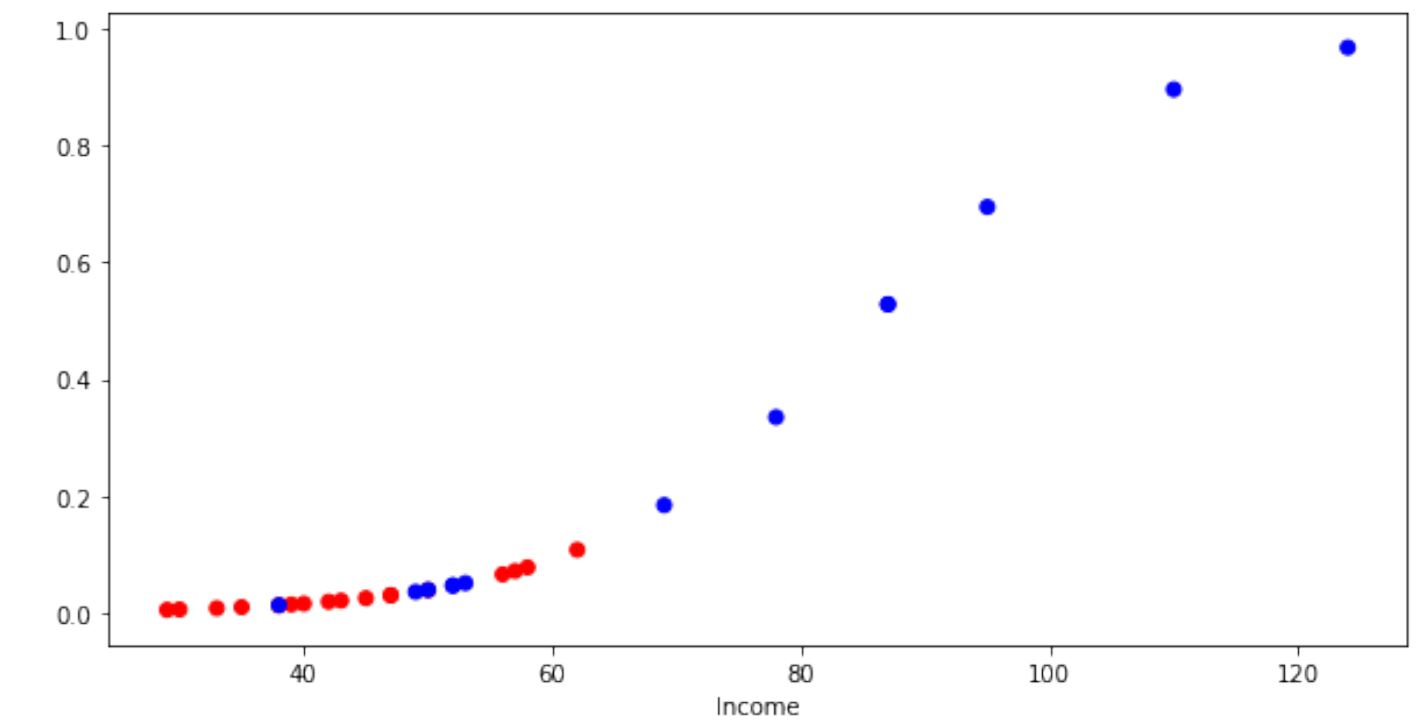
The cross entropy loss

- For each example:
 - If the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
 - If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$
- Then we average the cost over each example

$$CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N cost_i$$

- Why it might work?

$$V_{model} = \sigma(score)$$



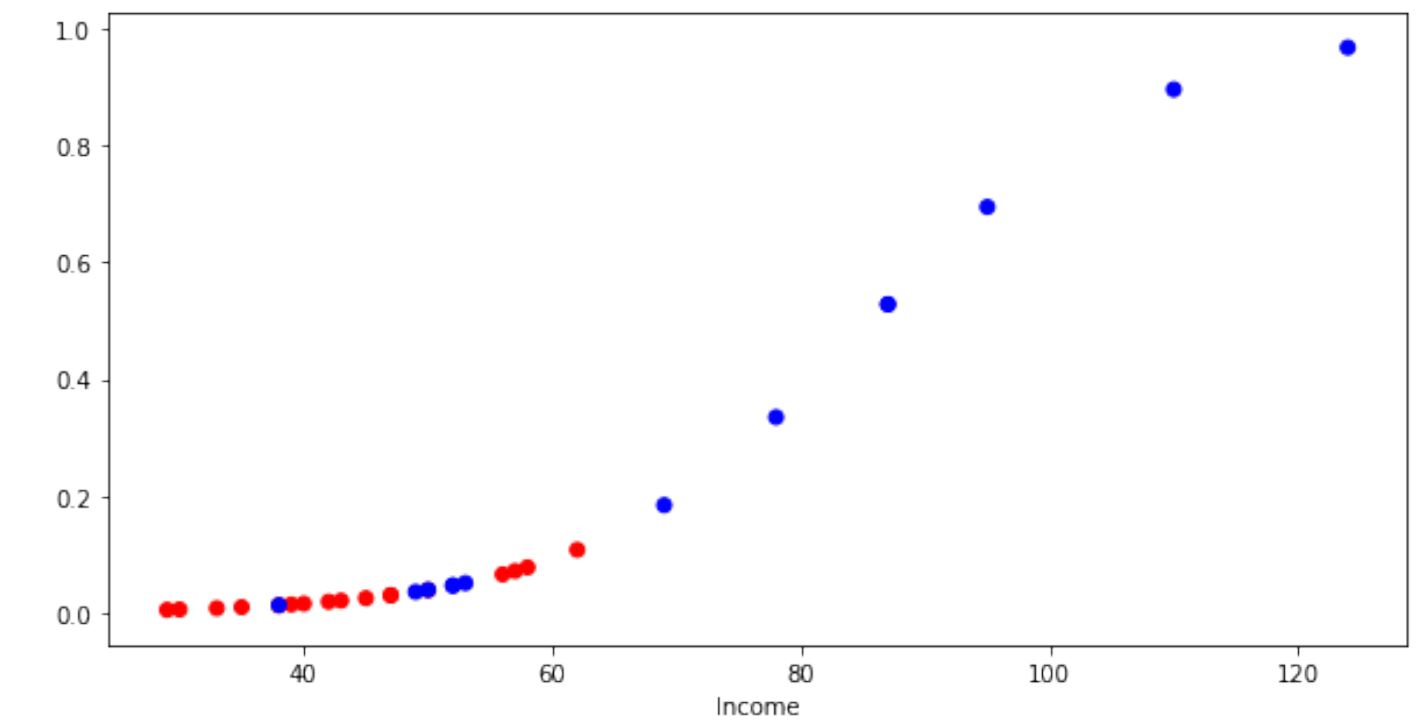
The cross entropy loss

- For each example:
 - If the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
 - If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$
- Then we average the cost over each example

$$CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N cost_i$$

- Why it might work?

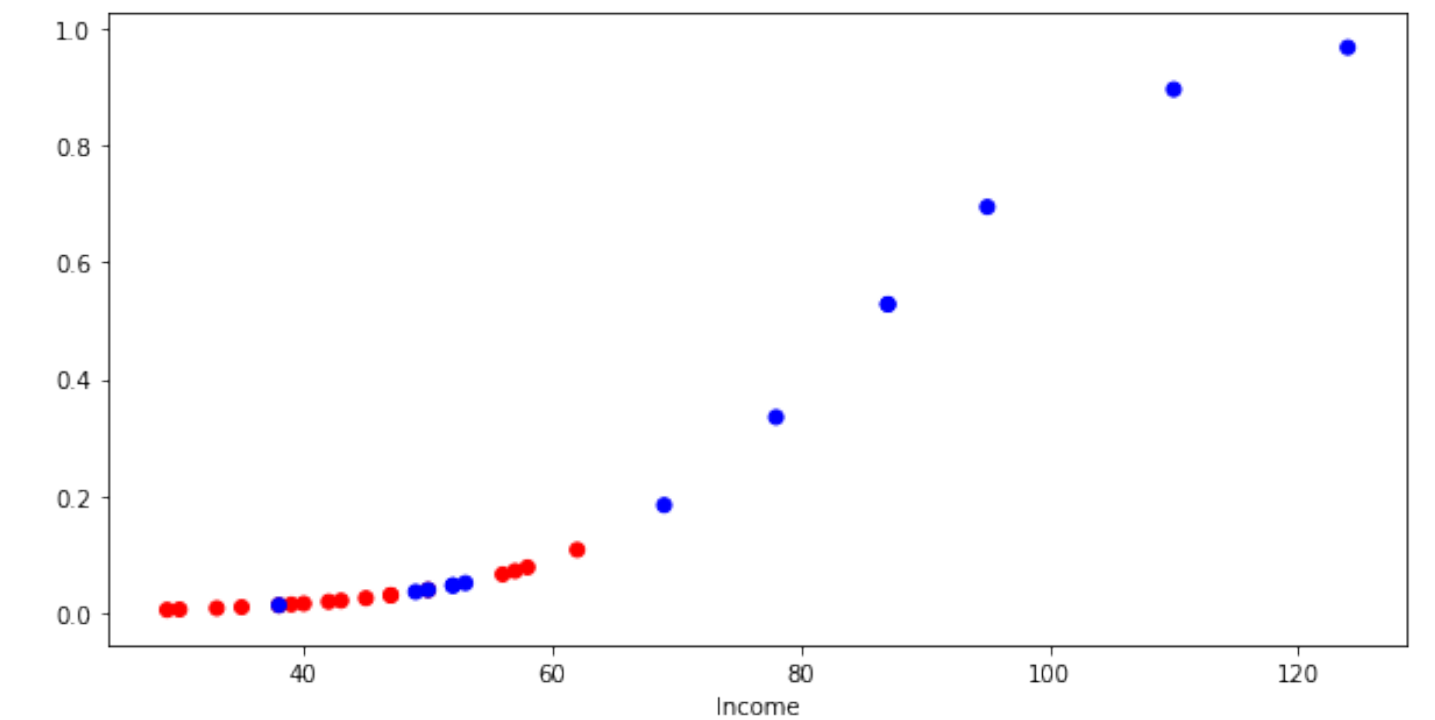
$$V_{model} = \sigma(score)$$



The cross entropy loss

- For each example:
 - If the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
 - If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$
- Then we average the cost over each example

$$V_{model} = \sigma(score)$$

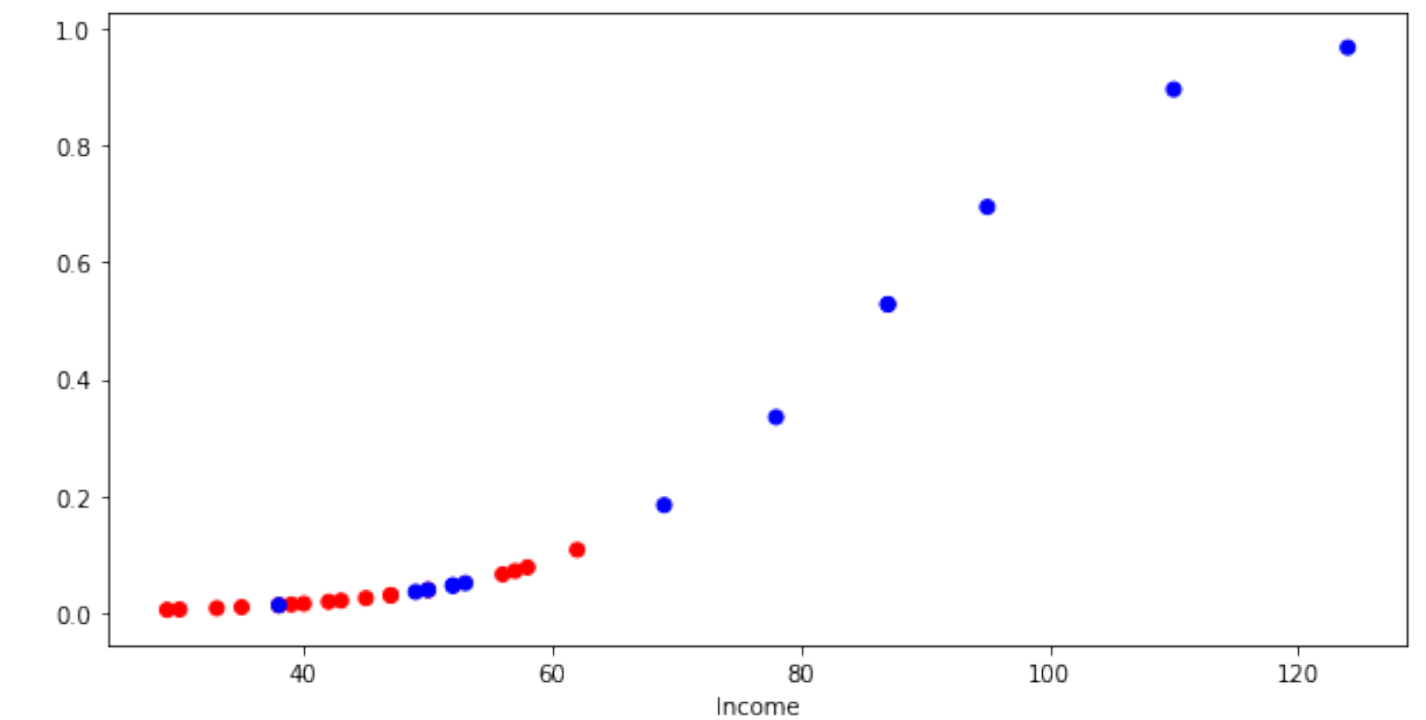


- If, for a given person, $\mathbf{V}=1$: the model say that this person will certainly vote for the right wing party
 - If the given person actually voted for the right-wing party (correct class is **1**), then the cost is $-\log(1) = 0$
 - If the given person actually voted for the left-wing party (correct class is **0**), then the cost is $-\log(1-1) = \infty$

The cross entropy loss

- For each example:
 - If the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
 - If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$
- Then we average the cost over each example

$$V_{model} = \sigma(score)$$

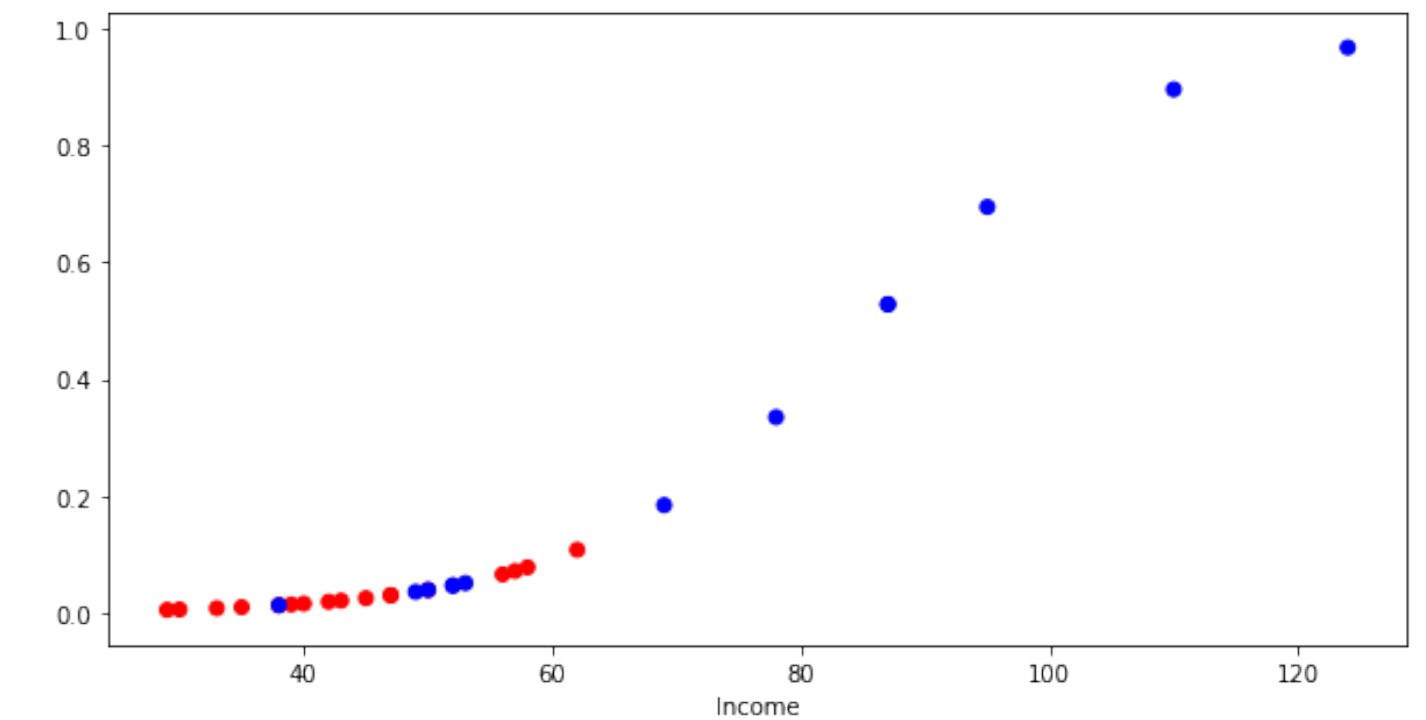


- If, for a given person, **V=0**: the model say that this person will certainly vote for the left wing party
 - If the given person actually voted for the right-wing party (correct class is **1**), then the cost is $-\log(\mathbf{0}) = \infty$
 - If the given person actually voted for the left-wing party (correct class is **0**), then the cost is $-\log(\mathbf{1}-\mathbf{0}) = 0$

The cross entropy loss

- For each example:
 - If the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
 - If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$
- Then we average the cost over each example

$$V_{model} = \sigma(score)$$



- If, for a given person, **V=0.5**: the model say that there is an equal probability that the voter will vote for any party
 - If the given person actually voted for the right-wing party (correct class is **1**), then the cost is $-\log(\mathbf{0.5}) = 0.69$
 - If the given person actually voted for the left-wing party (correct class is **0**), then the cost is $-\log(\mathbf{1-0.5}) = 0.69$

The cross entropy loss

- In short, the cost will be zero only if the model made a **very confident** correct prediction
- The cost will be very large if the model made a **very confident** incorrect prediction
- If the model made prediction with little confidence (V is close to 0.5), the cost will be moderate whatever the correct answer was
- If the model always make correct predictions with high confidence, the average of the cost will be close to zero

$$CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N cost_i$$

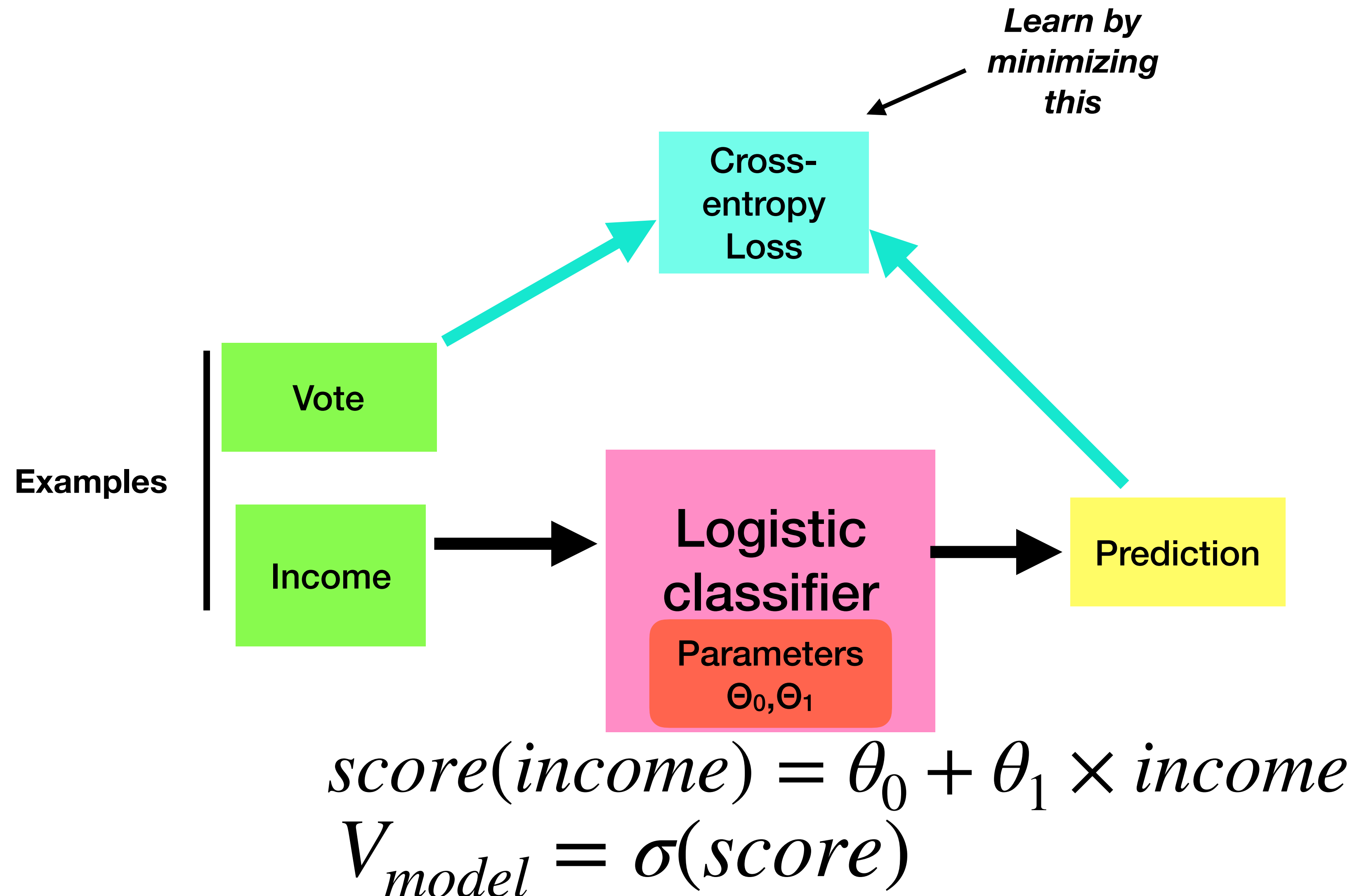
The cross entropy loss

- From this it follows that the `CrossEntropyLoss` is a good loss for our setting
- If we minimize it, we will find a good model
- It is a continuous (and smooth) function of the parameters
- It is easy to minimize in practice

$$\textit{CrossEntropyLoss} = \frac{1}{N} \sum_{i=1}^N \textit{cost}_i$$

Supervised Learning

- In supervised learning, we usually have:
 - A **MODEL**: a “parameterized” function that takes input and produce output
 - A **Loss**: A function that compute how different the model output is from the correct output
 - **Examples** of input and correct output (cigaretts smoked, age of death)



Minimizing the cross-entropy loss

- As before, to minimize the loss, we need to compute its gradient with respect to the parameters

$$score(income) = \theta_0 + \theta_1 \times income$$

$$V_{model} = \sigma(score)$$

$$CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N cost_i$$

We can find that:

$$\frac{\partial}{\partial \theta_0} CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N (V_{model}(income_i) - V_i)$$

$$\frac{\partial}{\partial \theta_1} CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N (V_{model}(income_i) - V_i) \times income_i$$

Minimizing the cross-entropy loss

- As before, to minimize the loss, we need to compute its gradient with respect to the parameters

$$score(income) = \theta_0 + \theta_1 \times income$$

We can find that:

$$V_{model} = \sigma(score)$$

$$CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N cost_i$$

$$\frac{\partial}{\partial \theta_0} CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N error_i$$

$$\frac{\partial}{\partial \theta_1} CrossEntropyLoss = \frac{1}{N} \sum_{i=1}^N error_i \times income_i$$

With:

$$error_i = V_{model}(income_i) - V_i$$

Our data

	income	vote
0	39.0	0
1	30.0	0
2	47.0	0
3	69.0	1
4	52.0	1
5	110.0	1
6	...	

$$grad \cdot CEL = [\frac{\partial}{\partial \theta_0} CEL, \frac{\partial}{\partial \theta_1} CEL] = [\frac{1}{N} \sum_{i=1}^N error_i, \frac{1}{N} \sum_{i=1}^N error_i \times income_i]$$

Minimizing the cross-entropy loss

- Now that we know the gradient of the loss, we can minimize the loss to find the best parameters θ
- To make things a bit less abstract, let's try to do it in practice on a small example

Example

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Suppose these parameters: $\theta_0 = -8$ $\theta_1 = 0.1$

$$\text{score}(\text{income}) = \theta_0 + \theta_1 \times \text{income}$$

$$V_{\text{model}} = \sigma(\text{score}) \qquad \sigma(x) = \frac{1}{1 + \exp(-x)}$$

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0				
2	70	1				
3	20	0				

Example

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Suppose these parameters: $\theta_0 = -8$ $\theta_1 = 0.1$

$$\text{score}(\text{income}) = \theta_0 + \theta_1 \times \text{income}$$

$$V_{\text{model}} = \sigma(\text{score}) \qquad \sigma(x) = \frac{1}{1 + \exp(-x)}$$

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0	-4	0.018	0	
2	70	1	-1	0.269	0	
3	20	0	-6	0.002	0	

Example

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Suppose these parameters: $\theta_0 = -8$ $\theta_1 = 0.1$

$$\text{score}(\text{income}) = \theta_0 + \theta_1 \times \text{income}$$

$$V_{\text{model}} = \sigma(\text{score})$$

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

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0	-4	0.018	0	
2	70	1	-1	0.269	0	
3	20	0	-6	0.002	0	

- if the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
- If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$

$$\text{CrossEntropyLoss} = \frac{1}{N} \sum_{i=1}^N \text{cost}_i$$

Example

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Suppose these parameters: $\theta_0 = -8$ $\theta_1 = 0.1$

$$\text{score}(\text{income}) = \theta_0 + \theta_1 \times \text{income}$$

$$V_{\text{model}} = \sigma(\text{score}) \qquad \sigma(x) = \frac{1}{1 + \exp(-x)}$$

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0	-4	0.018	0	0.018
2	70	1	-1	0.269	0	1.31
3	20	0	-6	0.002	0	0.002

- if the correct class is **1**, then the **cost** of the prediction **V** is $-\log(\mathbf{V})$
- If the correct class is **0**, then the **cost** of the prediction **V** is $-\log(1-\mathbf{V})$

$$\text{CrossEntropy} = \frac{1}{3}(0.018 + 1.31 + 0.002) = 0.44$$

Example

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

$$\theta_0 = -8 \quad \theta_1 = 0.1$$

- Now suppose we want to improve our parameters
- Let us do one iteration of gradient descent

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0	-4	0.018	0	0.018
2	70	1	-1	0.269	0	1.31
3	20	0	-6	0.002	0	0.002

$$CrossEntropy = \frac{1}{3}(0.018 + 1.31 + 0.002) = 0.44$$

Computing the gradient

$$\theta_0 = -8 \quad \theta_1 = 0.1$$

	income	vote	Score	Prediction V	Predicted Class	Cost	Error
1	40	0	-4	0.018	0	0.018	
2	70	1	-1	0.269	0	1.31	
3	20	0	-6	0.002	0	0.002	

$$error_i = V_{model}(income_i) - V_i$$

$$grad \cdot CEL = [\frac{\partial}{\partial \theta_0} CEL, \frac{\partial}{\partial \theta_1} CEL] = [\frac{1}{N} \sum_{i=1}^N error_i, \frac{1}{N} \sum_{i=1}^N error_i \times income_i]$$

Computing the gradient

$$\theta_0 = -8 \quad \theta_1 = 0.1$$

	income	vote	Score	Prediction V	Predicted Class	Cost	Error
1	40	0	-4	0.018	0	0.018	0.018
2	70	1	-1	0.269	0	1.31	-0.731
3	20	0	-6	0.002	0	0.002	0.002

$$error_i = V_{model}(income_i) - V_i$$

$$\begin{aligned} grad \cdot CEL &= \left[\frac{1}{N} \sum_{i=1}^N error_i, \frac{1}{N} \sum_{i=1}^N error_i \times income_i \right] \\ &= \frac{1}{3} [0.018 - 0.731 + 0.002, 0.018 \times 40 - 0.731 \times 70 + 0.002 \times 20] = [-0.23, -16.8] \end{aligned}$$

Doing one iteration of gradient descent

	income	vote	Score	Prediction V	Predicted Class	Cost	Error
1	40	0	-4	0.018	0	0.018	0.018
2	70	1	-1	0.269	0	1.31	-0.731
3	20	0	-6	0.002	0	0.002	0.002

$$\theta_0 = -8 \quad \theta_1 = 0.1$$

$$\text{grad} \cdot \text{CrossEntropy} = [-0.23, -16.8]$$

- Let us do one iteration of gradient descent with $\text{lr} = 0.005$
- What are new parameters θ ?

Doing one iteration of gradient descent

	income	vote	Score	Prediction V	Predicted Class	Cost	Error
1	40	0	-4	0.018	0	0.018	0.018
2	70	1	-1	0.269	0	1.31	-0.731
3	20	0	-6	0.002	0	0.002	0.002

$$\theta_0 = -8 \quad \theta_1 = 0.1$$

$$\text{grad} \cdot \text{CrossEntropy} = [-0.23, -16.8]$$

- Let us do one iteration of gradient descent with $\text{lr} = 0.005$
- What are new parameters θ ?

$$\theta_0 = -8 - 0.005 \times -0.23 = -7.999$$

$$\theta_1 = 0.1 - 0.005 \times -16.8 = 0.184$$

Trying the new parameters

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Let us try these new parameters

$$\theta_0 = -7.999$$

$$\theta_1 = 0.184$$

$$score(income) = \theta_0 + \theta_1 \times income$$

$$V_{model} = \sigma(score)$$

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

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0				
2	70	1				
3	20	0				

Trying the new parameters

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Let us try these new parameters

$$\theta_0 = -7.999$$

$$\theta_1 = 0.184$$

$$score(income) = \theta_0 + \theta_1 \times income$$

$$V_{model} = \sigma(score)$$

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

	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0	-0.639	0.34	0	
2	70	1	4.88	0.99	1	
3	20	0	-4.319	0.01	0	

Trying the new parameters

Example Data

	income	vote
1	40	0
2	70	1
3	20	0

- Let us try these new parameters

$$\theta_0 = -7.999$$

$$\theta_1 = 0.184$$

$$score(income) = \theta_0 + \theta_1 \times income$$

$$V_{model} = \sigma(score)$$

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

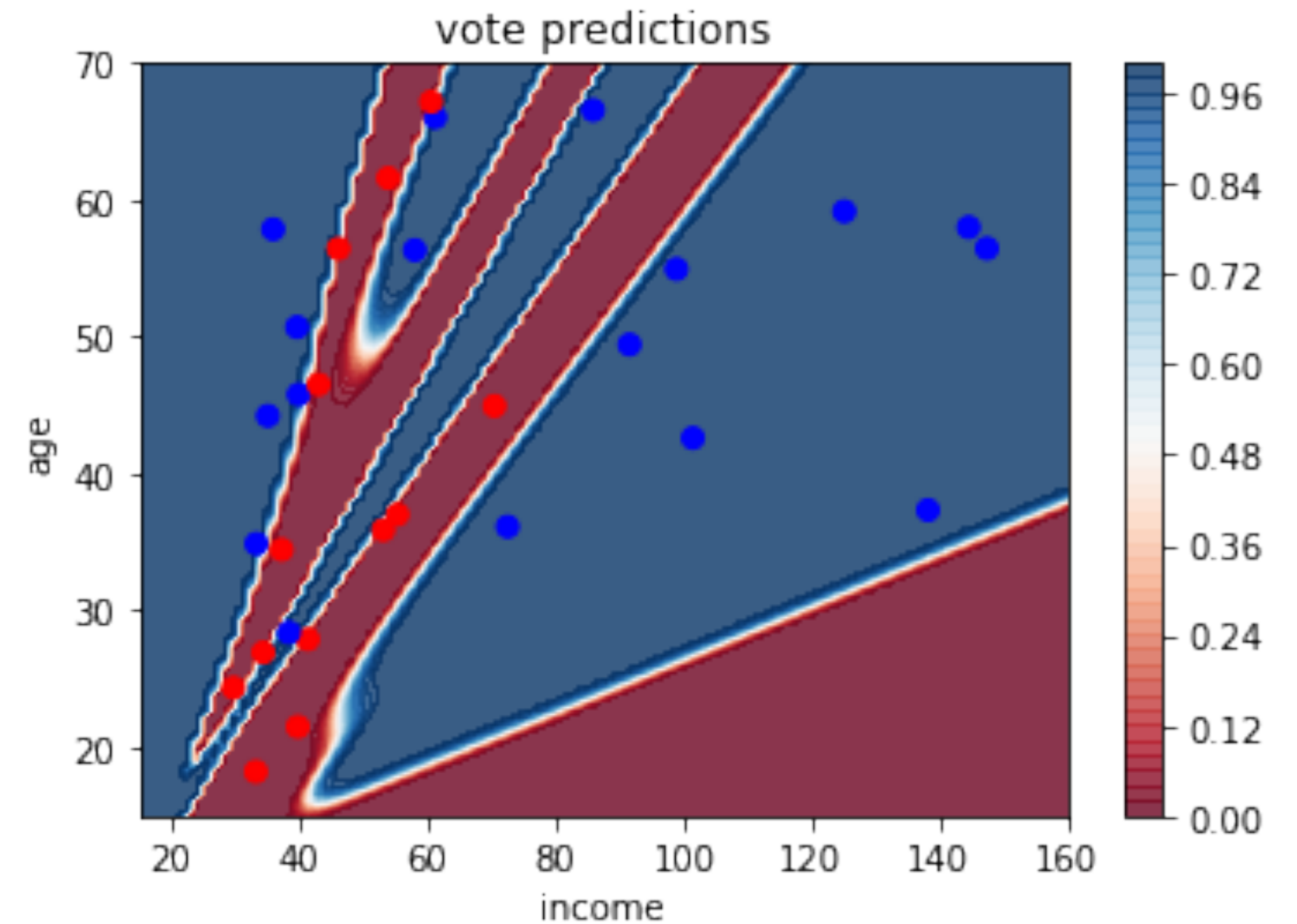
	income	vote	Score	Prediction V	Predicted Class	Cost
1	40	0	-0.639	0.34	0	0.4
2	70	1	4.88	0.99	1	0.01
3	20	0	-4.319	0.01	0	0.013

$$CrossEntropy = \frac{1}{3}(0.4 + 0.01 + 0.013) = 0.15$$

Cross entropy decreased from 0.43 to 0.15

**Beyond binary
classification with
one feature:**

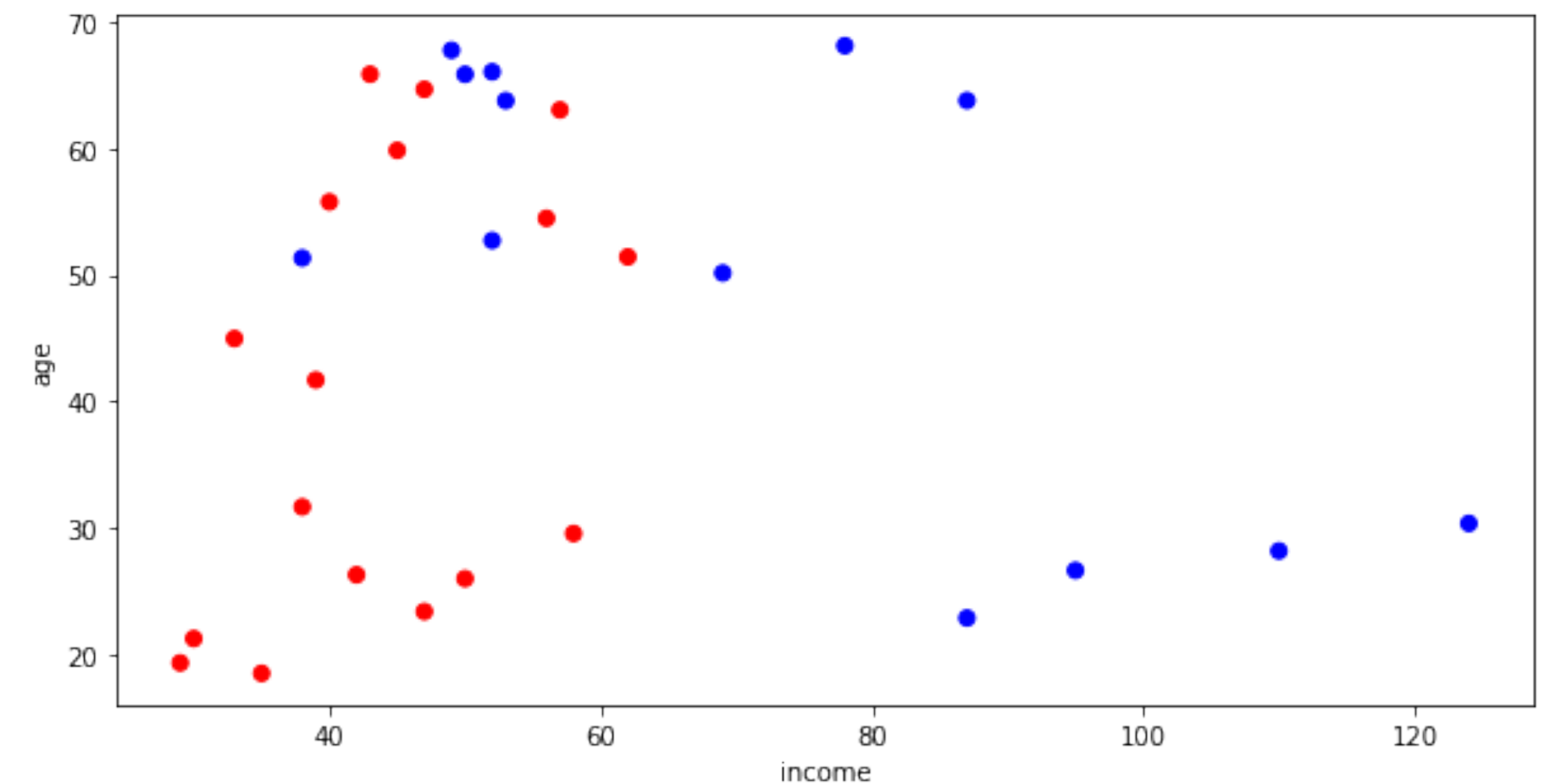
**Many features,
complex models**



Classification with more than one feature

- Let us now suppose that, on top of **income**, we have information about **age**
- How do we change our Logistic model to take into account the new information?

	income	age	vote
0	39.0	42.0	L
1	30.0	21.0	L
2	47.0	65.0	L
3	69.0	50.0	R
4	52.0	53.0	R
5	110.0	28.0	R
...	...	...	...



Classification with more than one feature

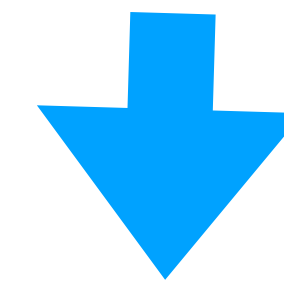
- Let us now suppose that, on top of **income**, we have information about **age**
- How do we change our Logistic model to take into account the new information?

	income	age	vote
0	39.0	42.0	L
1	30.0	21.0	L
2	47.0	65.0	L
3	69.0	50.0	R
4	52.0	53.0	R
5	110.0	28.0	R
...	...	...	...

EASY! We just add a new parameter to the score function:

$$score(income) = \theta_0 + \theta_1 \times income$$

$$V_{model} = \sigma(score)$$



$$score(income, age) = \theta_0 + \theta_1 \times income + \theta_2 \times age$$

$$V_{model} = \sigma(score)$$

Classification with more than one feature

- Let us now suppose that, on top of **income**, we have information about **age**
- How do we change our Logistic model to take into account the new information?

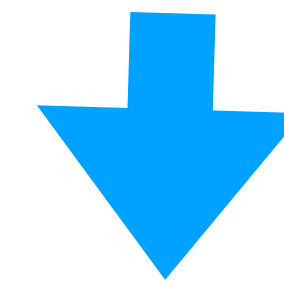
	income	age	vote
0	39.0	42.0	L
1	30.0	21.0	L
2	47.0	65.0	L
3	69.0	50.0	R
4	52.0	53.0	R
5	110.0	28.0	R
...	...	...	...

Nothing else to change:
Everything we have seen about cost, loss,
gradient descent work the same

EASY! We just add a new parameter to the score function:

$$score(income) = \theta_0 + \theta_1 \times income$$

$$V_{model} = \sigma(score)$$

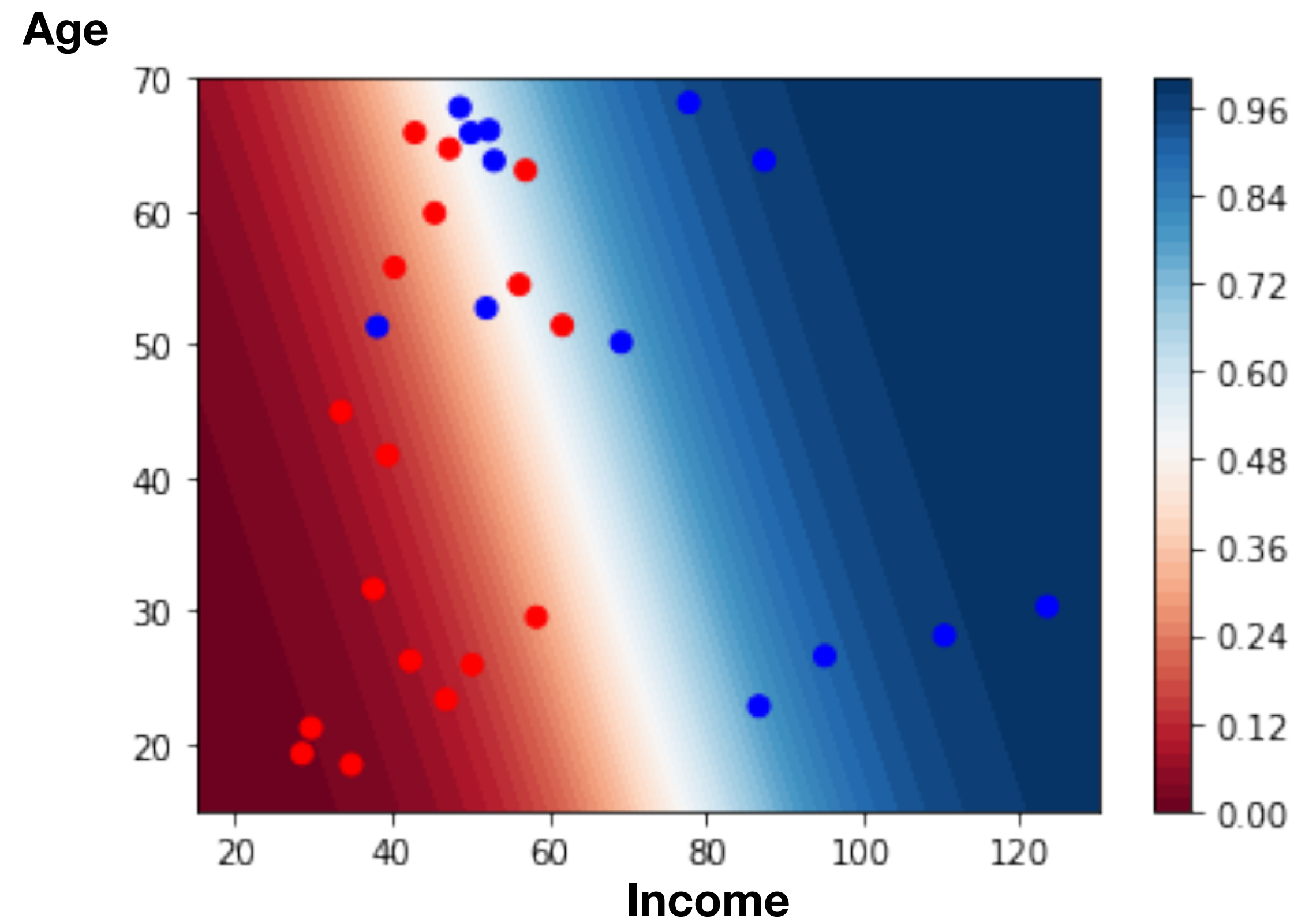


$$score(income, age) = \theta_0 + \theta_1 \times income + \theta_2 \times age$$

$$V_{model} = \sigma(score)$$

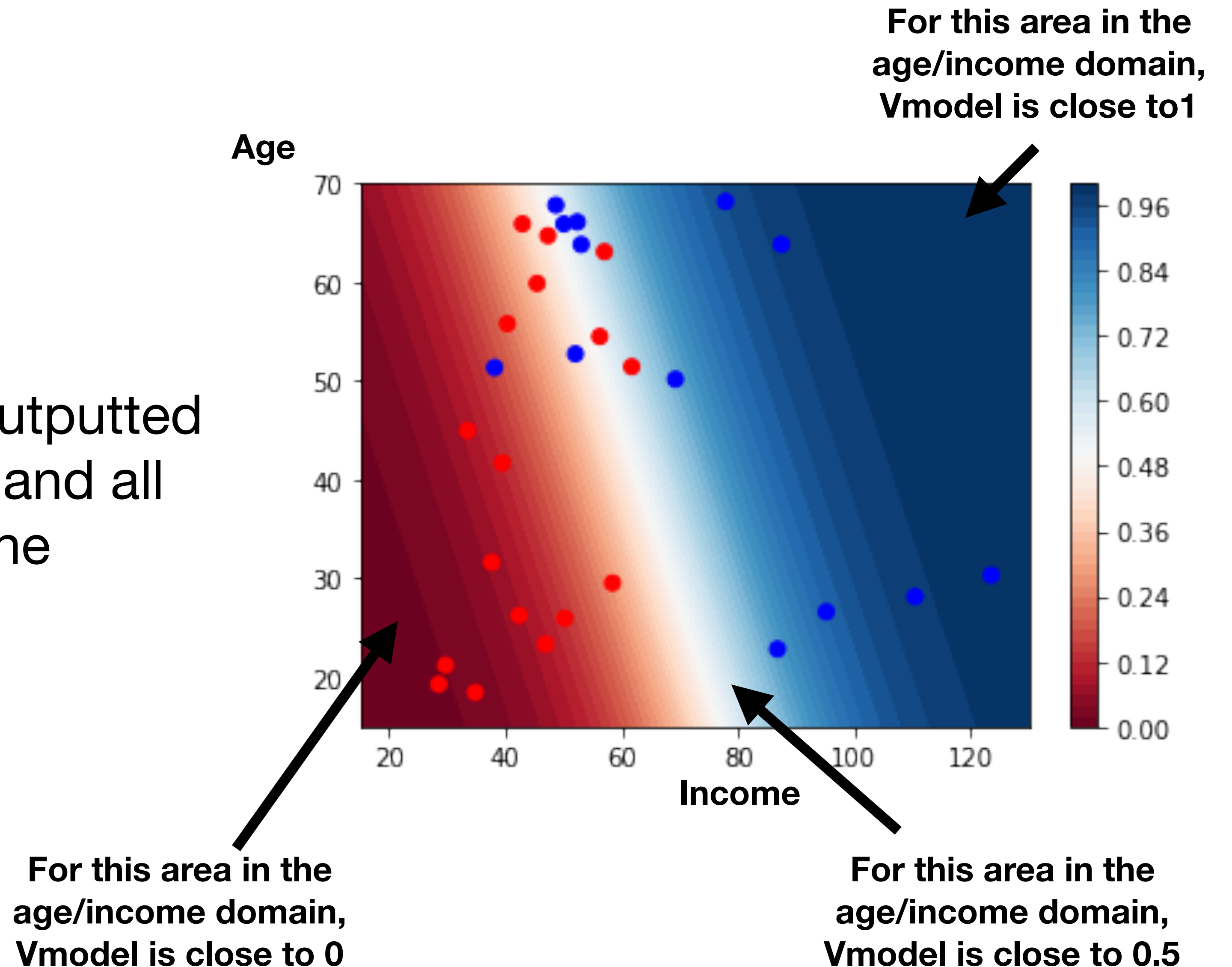
Visualizing the decision boundary

- If we represent the value outputted by the classifier for all age and all income, we can visualize the boundary decision



Visualizing the decision boundary

- If we represent the value outputted by the classifier for all age and all income, we can visualize the boundary decision



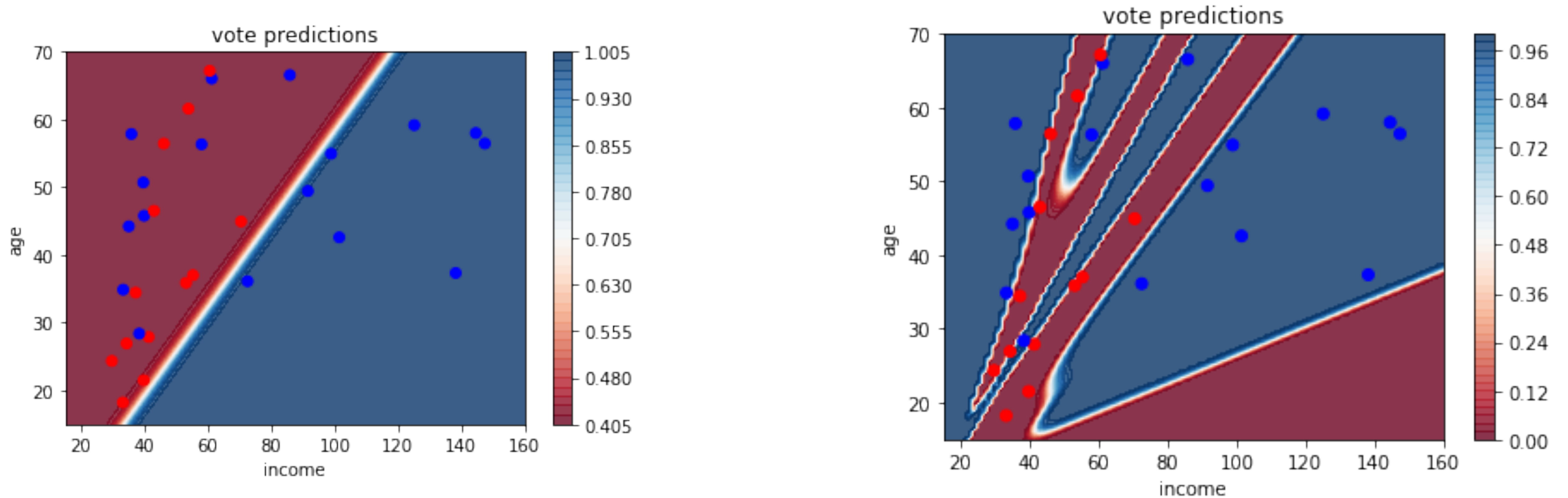
More complex models

- In fact, the score that we compute is like the linear models that we were using for linear regression.
- Therefore, we can apply the same tricks in term of feature expansion to learn a more complex model

$$\begin{aligned} \text{score}(\text{income}, \text{age}) = & \theta_0 + \theta_1 \times \text{income} \\ & + \theta_2 \times \text{age} + \theta_3 \times \text{income}^2 \\ & + \theta_4 \times \text{age}^2 + \theta_5 \times \text{income} \times \log(\text{age}) + \dots \end{aligned}$$

- But beware overfitting

Overfitting



$$\text{score}(\text{income}, \text{age}) = \theta_0 + \theta_1 \times \text{income} + \theta_2 \times \text{age}$$

$$\begin{aligned} \text{score}(\text{income}, \text{age}) = & \theta_0 + \theta_1 \times \text{income} \\ & + \theta_2 \times \text{age} + \theta_3 \times \text{income}^2 \\ & + \theta_4 \times \text{age}^2 + \theta_5 \times \text{income} \times \log(\text{age}) + \dots \end{aligned}$$

$$V_{\text{model}} = \sigma(\text{score})$$

Multiclass Classification



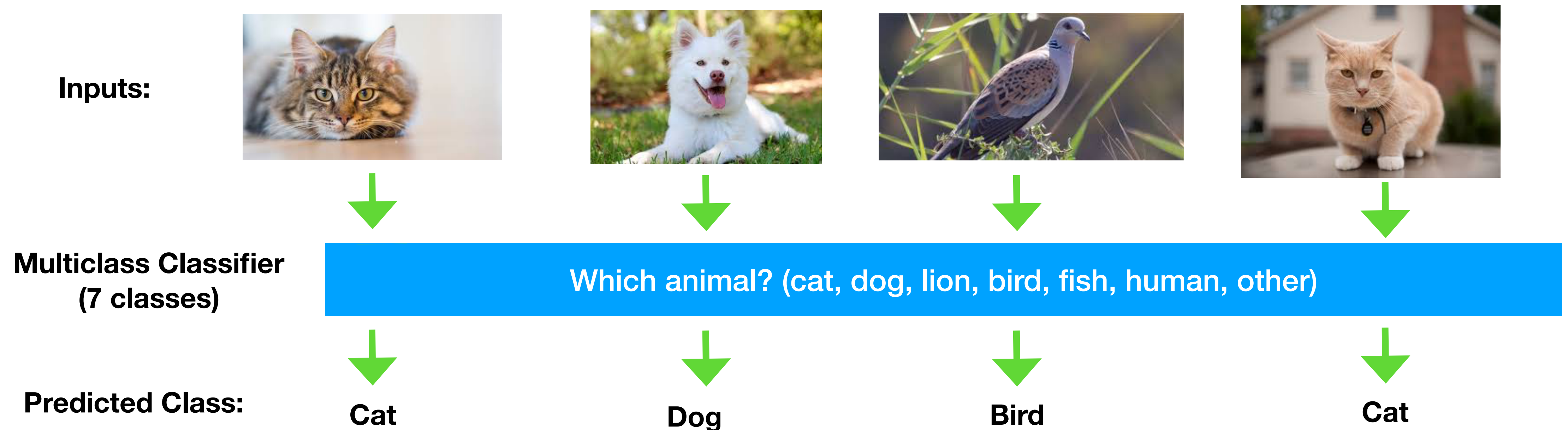
Which animal? (cat, dog, lion, bird, fish,
human, other)



Cat

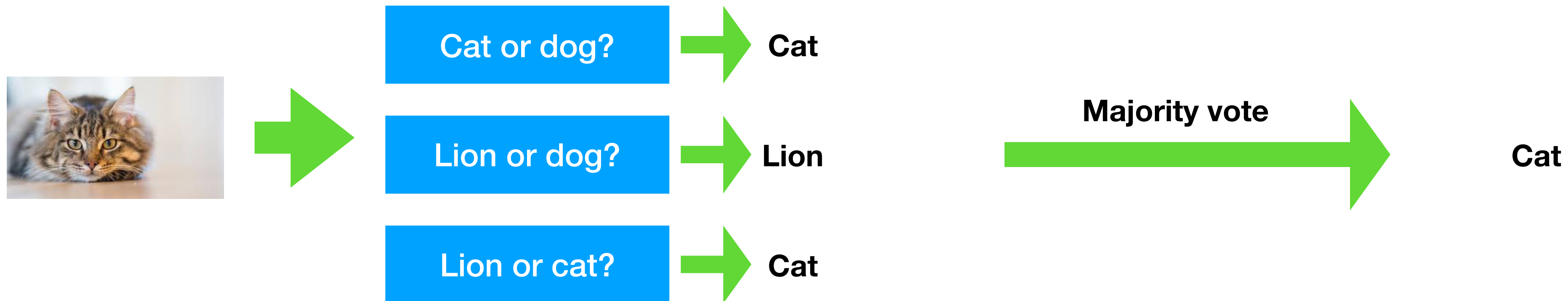
Multiclass Classification

- Remember that multi class classification is when we have more than 2 possible answers



Multiclass Classification

- How do we do multi class classification?
- Mainly two options:
 - 1- Reduce to binary classification
 - e.g. One vs One approach: train a binary classifier for each pair of category



Multiclass Classification

- How do we do multi class classification?
- Mainly two options:
 - 1- Reduce to binary classification
 - e.g. One vs One approach: train a binary classifier for each pair of category
 - 2- Use multinomial Logistic Regression (a.k.a **Softmax** classification)

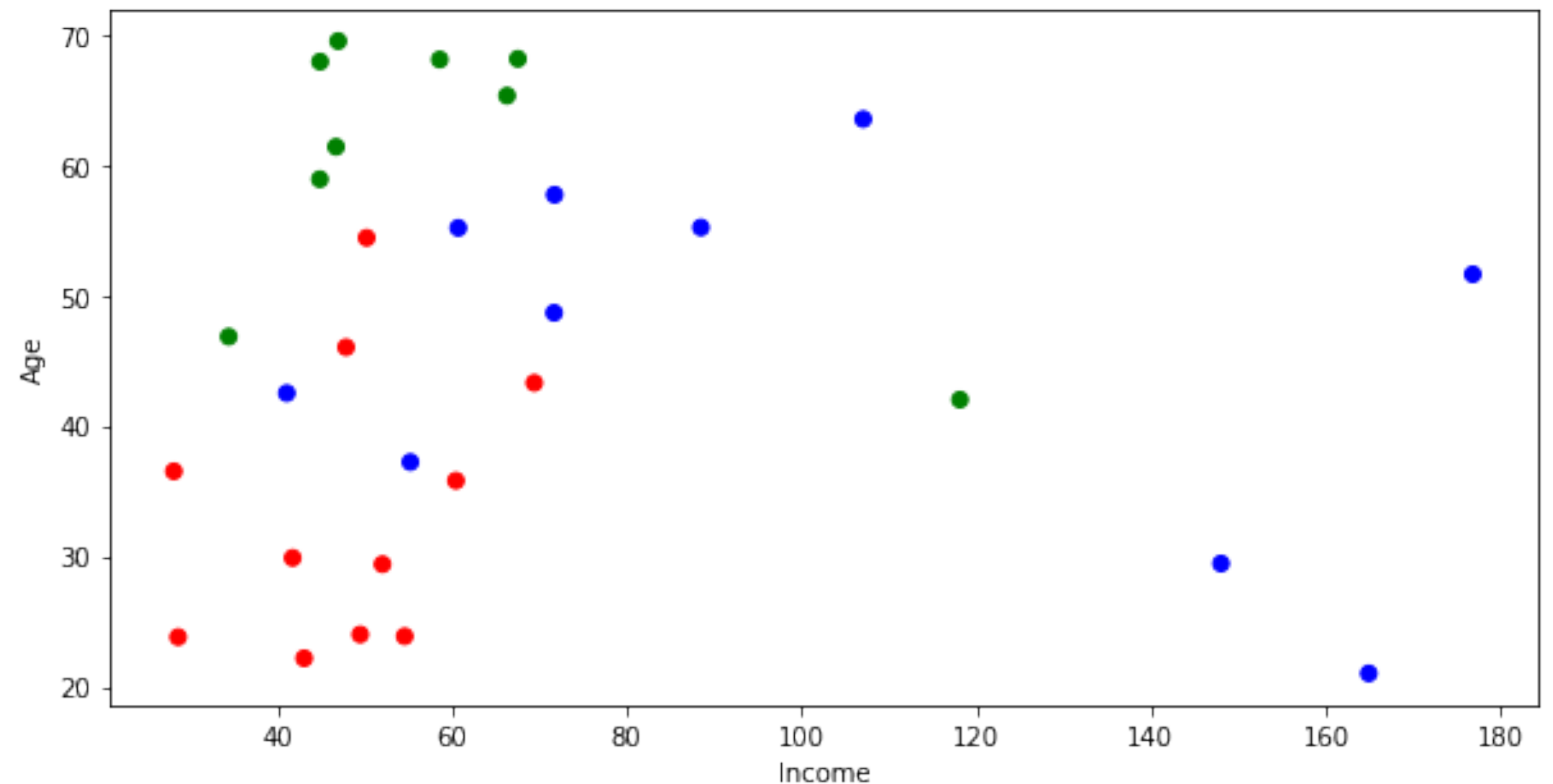
Multiclass Classification

- How do we do multi class classification?
- Mainly two options:
 - 1- Reduce to binary classification
 - e.g. One vs One approach: train a binary classifier for each pair of category
 - 2- Use multinomial Logistic Regression (a.k.a **Softmax** classification)
 - Similar to Binary Logistic Classification
 - But we are going to compute a **score** function for each of the classes

Multiclass Classification

- Let us now suppose that we have 3 parties: Left-Wing, Right-Wing and “Green” party
- How are we going to predict the votes with Softmax classification?

	income	age	vote
0	48.0	46.0	L
1	72.0	58.0	R
2	72.0	49.0	R
3	29.0	24.0	L
4	47.0	70.0	G
5	177.0	52.0	R
6	...	...	...



Multiclass Classification

- Let us now suppose that we have 3 parties: Left-Wing, Right-Wing and “Green”
- How are we going to predict the votes with Softmax classification?

	income	age	vote
0	48.0	46.0	L
1	72.0	58.0	R
2	72.0	49.0	R
3	29.0	24.0	L
4	47.0	70.0	G
5	177.0	52.0	R
6	...	...	...

$$score_L(income, age) = \theta_0^L + \theta_1^L \times income + \theta_2^L \times age$$

$$score_R(income, age) = \theta_0^R + \theta_1^R \times income + \theta_2^R \times age$$

$$score_G(income, age) = \theta_0^G + \theta_1^G \times income + \theta_2^G \times age$$

We now have $3 \times 3 = 9$ parameters

$$V_{model} = softmax(score_L, score_R, score_G)$$

The softmax function

- The **softmax** function (sometimes called *multinomial logistic function*) is used for multiclass classification
- The input of a softmax function is a vector of N dimensions
- The output is a N-dimension vector that can represents probabilities for each class
- It is computed by taking the exponential of the components of a vector, then dividing by the sum of the exponentiated components

$$\text{softmax}(\vec{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

The softmax function

- Example: computing $\text{softmax}([1, -1, 0, 2])$

$$[1, -1, 0, 2] \xrightarrow{\text{1- Exponential}} [e^1, e^{-1}, e^0, e^2] \approx [2.71, 0.36, 1, 7.39]$$

3- Divide
by sum

2- Sum:

$$2.71 + 0.36 + 1 + 7.39 = 11.46$$

$$\left[\frac{2.71}{11.46}, \frac{0.36}{11.46}, \frac{1}{11.46}, \frac{7.39}{11.46} \right] = [0.23, 0.03, 0.09, 0.65]$$

Softmax

The softmax function

- Example: computing $\text{softmax}([1, -1, 0, 2])$

$$[1, -1, 0, 2] \xrightarrow{\text{1- Exponential}} [e^1, e^{-1}, e^0, e^2] \approx [2.71, 0.36, 1, 7.39]$$

**This step
makes all
values > 0**

**3- Divide
by sum**

2- Sum:

$$2.71 + 0.36 + 1 + 7.39 = 11.46$$

$$\left[\frac{2.71}{11.46}, \frac{0.36}{11.46}, \frac{1}{11.46}, \frac{7.39}{11.46} \right] = [0.23, 0.03, 0.09, 0.65]$$

**This step makes all
values sum to 1**

Softmax

**Positive values
summing to 1:
Can be seen as
probabilities**

Multiclass Classification

- Let us now suppose that we have 3 parties: Left-Wing, Right-Wing and “Green”
- How are we going to predict the votes?

	income	age	vote
0	48.0	46.0	L
1	72.0	58.0	R
2	72.0	49.0	R
3	29.0	24.0	L
4	47.0	70.0	G
5	177.0	52.0	R
6	...	...	...

$$score_L(income, age) = \theta_0^L + \theta_1^L \times income + \theta_2^L \times age$$

$$score_R(income, age) = \theta_0^R + \theta_1^R \times income + \theta_2^R \times age$$

$$score_G(income, age) = \theta_0^G + \theta_1^G \times income + \theta_2^G \times age$$

$$V_{model} = softmax(score_L, score_R, score_G)$$

We now have $3 \times 3 = 9$ parameters

This is now a vector of dimension 3 that contains probabilities of voting for each party

Example

$$\theta_0^L = -3$$

$$\theta_1^L = 0.1$$

$$\theta_2^L = 0.2$$

$$score_L(income, age) = \theta_0^L + \theta_1^L \times income + \theta_2^L \times age$$

$$\theta_0^R = -5$$

$$\theta_1^R = 0.3$$

$$\theta_2^R = 0.4$$

$$score_R(income, age) = \theta_0^R + \theta_1^R \times income + \theta_2^R \times age$$

$$\theta_0^G = 0$$

$$\theta_1^G = 0.1$$

$$\theta_2^G = 0.1$$

$$score_G(income, age) = \theta_0^G + \theta_1^G \times income + \theta_2^G \times age$$

$$V_{model} = \text{softmax}(score_L, score_R, score_G)$$

Somebody is 40 year old, with income of 50

Which party is predicted by this model?

Example

$$\theta_0^L = -3 \quad \theta_1^L = 0.1 \quad \theta_2^L = 0.2 \quad \text{score}_L(\text{income}, \text{age}) = \theta_0^L + \theta_1^L \times \text{income} + \theta_2^L \times \text{age}$$

$$\theta_0^R = -5 \quad \theta_1^R = 0.3 \quad \theta_2^R = 0.4 \quad \text{score}_R(\text{income}, \text{age}) = \theta_0^R + \theta_1^R \times \text{income} + \theta_2^R \times \text{age}$$

$$\theta_0^G = 0 \quad \theta_1^G = 0.1 \quad \theta_2^G = 0.1 \quad \text{score}_G(\text{income}, \text{age}) = \theta_0^G + \theta_1^G \times \text{income} + \theta_2^G \times \text{age}$$

$$V_{\text{model}} = \text{softmax}(\text{score}_L, \text{score}_R, \text{score}_G)$$

Somebody is 40 year old, with income of 50

Which party is predicted by this model?

$$\text{score}_L(50, 40) = -3 + 0.1 \times 50 + 0.2 \times 40 = 15$$

$$\text{score}_R(50, 40) = -5 + 0.3 \times 50 + 0.4 \times 40 = 26$$

$$\text{score}_G(50, 40) = 0 + 0.1 \times 50 + 0.1 \times 40 = 9$$

**This model says he has 99.9%
chances of voting for the right wing
party**

$$\text{softmax}([15, 26, 9]) = \frac{1}{e^{15} + e^{26} + e^9} [e^{15}, e^{26}, e^9] = [0.000199, 0.9998, 0.000001]$$

Multiclass Classification

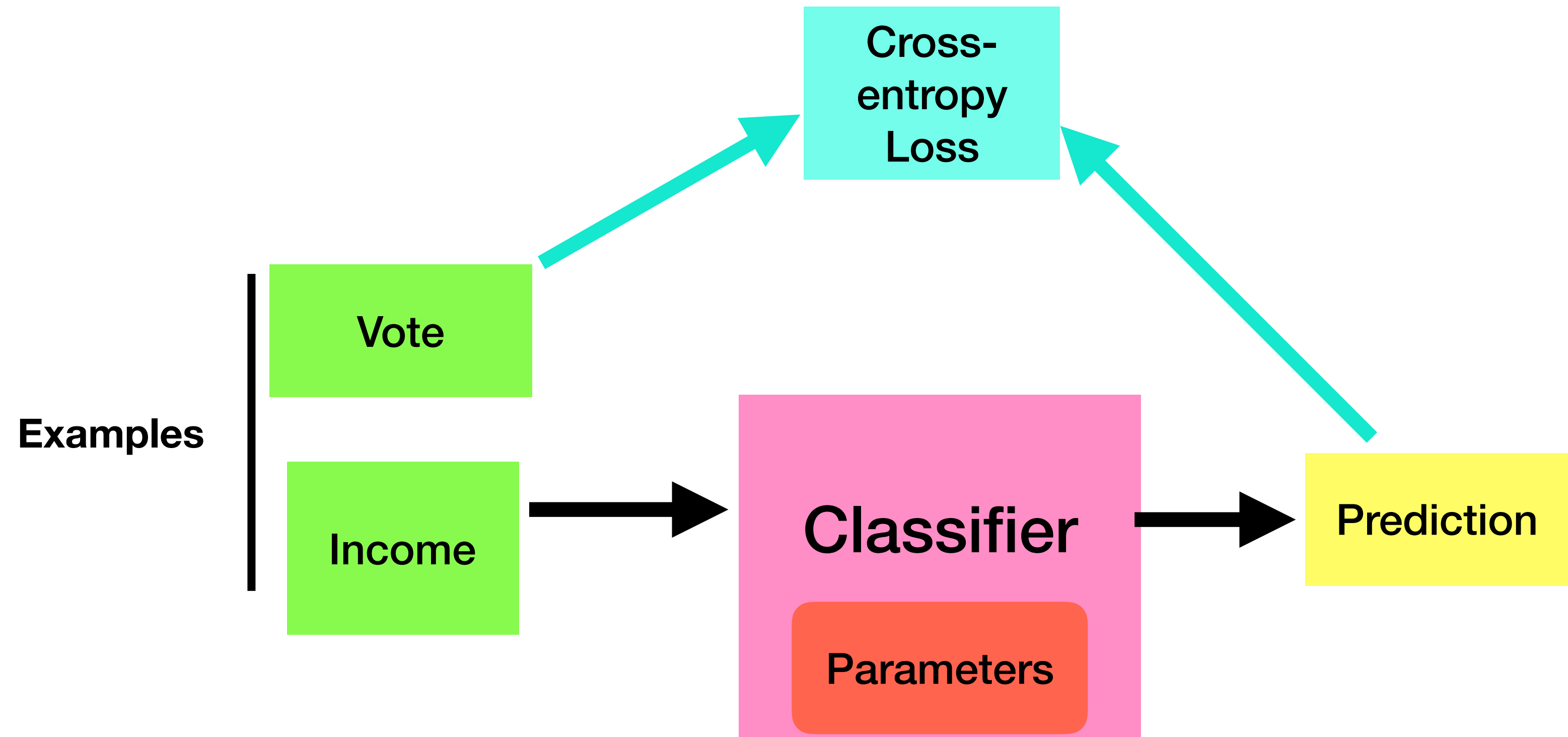
- Finding the optimal parameters is once again very similar to the case of binary classification
- We will use a cross-entropy loss again

Supervised Learning

Learn by
minimizing
this

- In supervised learning, we usually have:

- A **MODEL**: a “parameterized” function that takes input and produce output
- A **Loss**: A function that compute how different the model output is from the correct output
- **Examples** of input and correct output (cigarets smoked, age of death)



$$score_L(income, age) = \theta_0^L + \theta_1^L \times income + \theta_2^L \times age$$

$$score_R(income, age) = \theta_0^R + \theta_1^R \times income + \theta_2^R \times age$$

$$score_G(income, age) = \theta_0^G + \theta_1^G \times income + \theta_2^G \times age$$

$$V_{model} = softmax(score_L, score_R, score_G)$$

Multiclass Classification

- Finding the optimal parameters is once again very similar to the case of binary classification
- We will use a cross-entropy loss again (adapted for multi-class case)
- For each example:
 - If the correct class is i , then the **cost** of the prediction $-\log(\mathbf{V}[i])$
- Then we average the cost over each example

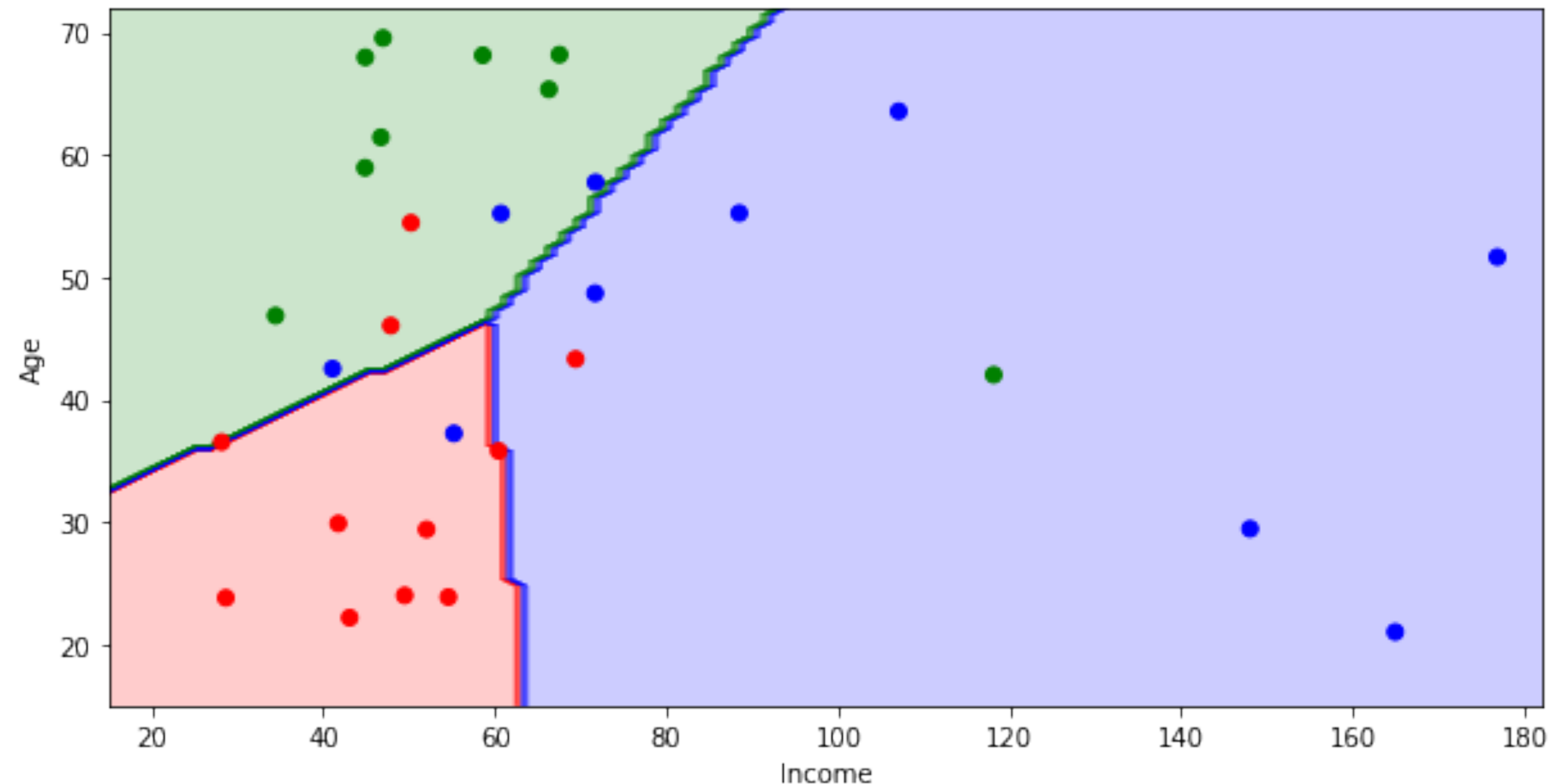
Visualization of boundary decisions

After learning good parameters, we can visualize the model that was learned.

The graph is colored according to the party that get the maximum score for a given age/income pair

We see the areas are separated by straight lines: this is because our score functions are linear functions of age and income

	income	age	vote
0	48.0	46.0	L
1	72.0	58.0	R
2	72.0	49.0	R
3	29.0	24.0	L
4	47.0	70.0	G
5	177.0	52.0	R
6	...	...	...



Summary

- We have seen the two most important type of tasks in supervised learning:
 - **Regression** (predict a value) : e.g. age of death
 - **Classification** (predict a category): e.g. Political party
- Using “simple” models:
 - **Linear Regression** for regression
 - **Logistic Regression** for classification
 - (Note: Logistic Regression should be called “Logistic Classification”, but it is a historical term)

Next time

- Next time, we will start discussing the fancy things:
 - **Neural Networks** and Deep Learning
- These will let us solve the same tasks of **classification** and **regression**, but with much more powerful models