Credit Risk Modeling with Machine Learning

What a real-world machine learning solution looks like — no background knowledge required

A. Jeremy Mahoney · Follow

Published in Towards Data Science · 22 min read · Sep 9, 2020

Note from Towards Data Science's editors: While we allow independent authors to publish articles in accordance with our rules and guidelines, we do not endorse each author's contribution. You should not rely on an author's works without seeking professional advice. See our Reader Terms for details.

Credit risk modeling–the process of estimating the probability someone will pay back a loan–is one of the most important mathematical problems of the modern world. In this article, we'll explore from the ground up how **machine learning** is applied to credit risk modeling. You don't need to know anything about machine learning to understand this article!

To explain credit risk modeling with machine learning, we'll first develop domain knowledge about credit risk modeling. Then, we'll introduce four fundamental machine learning systems that can be used for credit risk modeling:

- K-Nearest Neighbors
- Logistic Regression
- Decision Trees
- Neural Networks

By the end of this article, you'll understand how each of these algorithms can be applied to the real-world problem of credit risk modeling, and you'll be well on your way to understanding the field of machine learning in general!

Let's begin learning about what credit risk modeling is by looking at a simple situation.

The Situation

Say your buddy Ted needs ten bucks. You'll want those bucks back, so he promises he'll repay you tomorrow when you see him again.

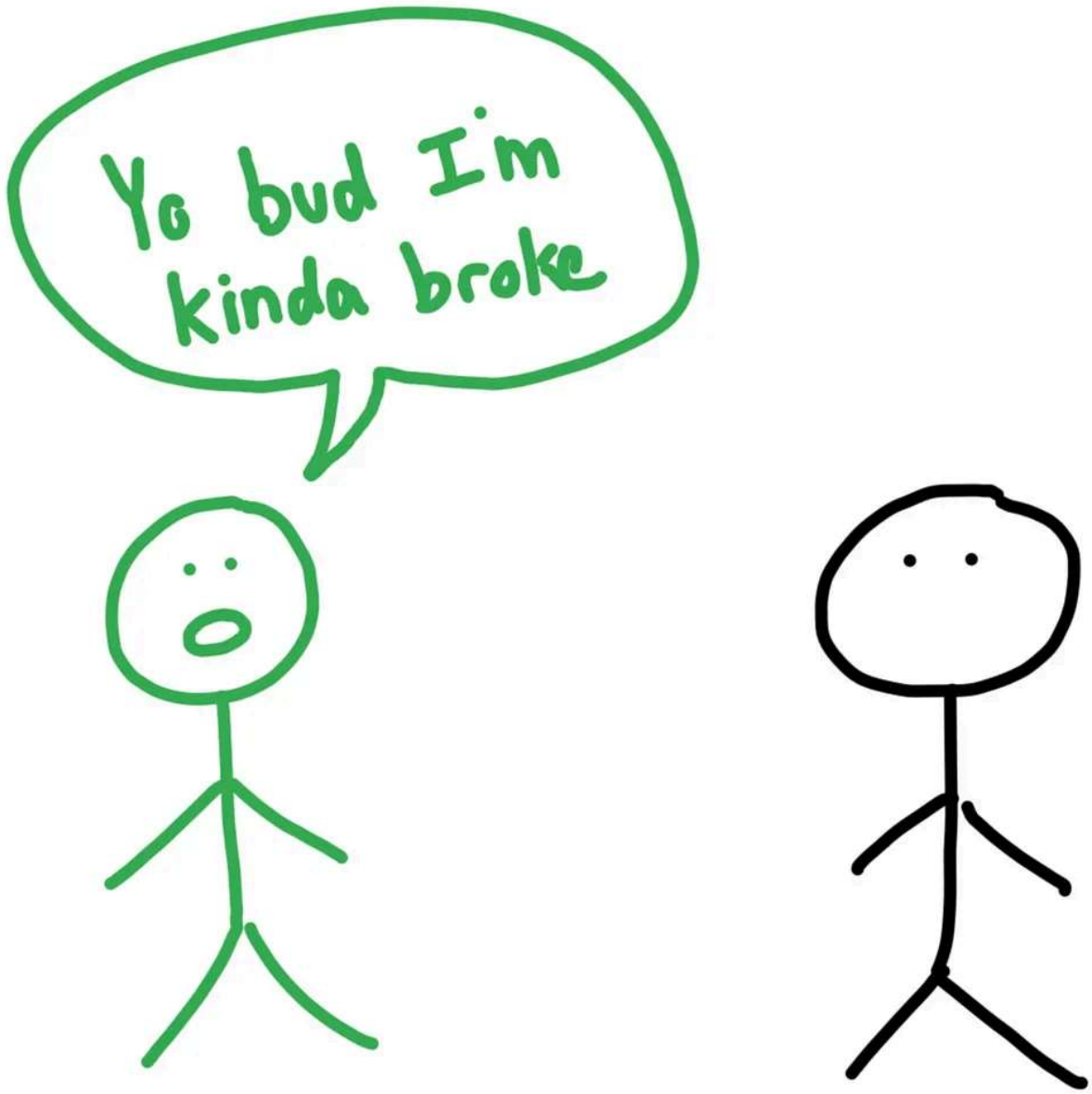

Image created by author

But you've heard that Ted has a bit of a history with being difficult when it comes to repaying people. As a result, you know there's a chance you'll never see your ten dollars again.

Would you lend him the money?

To make this decision, you need another piece of information.

Just *how* difficult has Ted been in the past?

If he just forgot to pay back the wrong dude once, then you're probably good. On the other hand, if he has a habit of getting deep into debt and fleeing to other countries, then you should probably keep your money to yourself.

Ted's history tells us something about the **risk** of lending money to him.

If he's been trustworthy in the past, then the **risk** of lending money to him is relatively low. If he's been untrustworthy in the past, then the **risk** of lending money to him is relatively high.

The concept of risk gives you a logical way to make the decision about whether or not to lend Ted the money. First, determine how much risk you're okay with. Then, figure out how risky Ted's history makes lending money to him. If the risk of lending money to Ted isn't greater than your maximum tolerated risk, then you can go ahead and lend him the money.

If it's just Ted asking you for money, you'll probably rely on your intuition and feelings to determine the risk associated with loaning money to Ted. This works well enough for just one person.

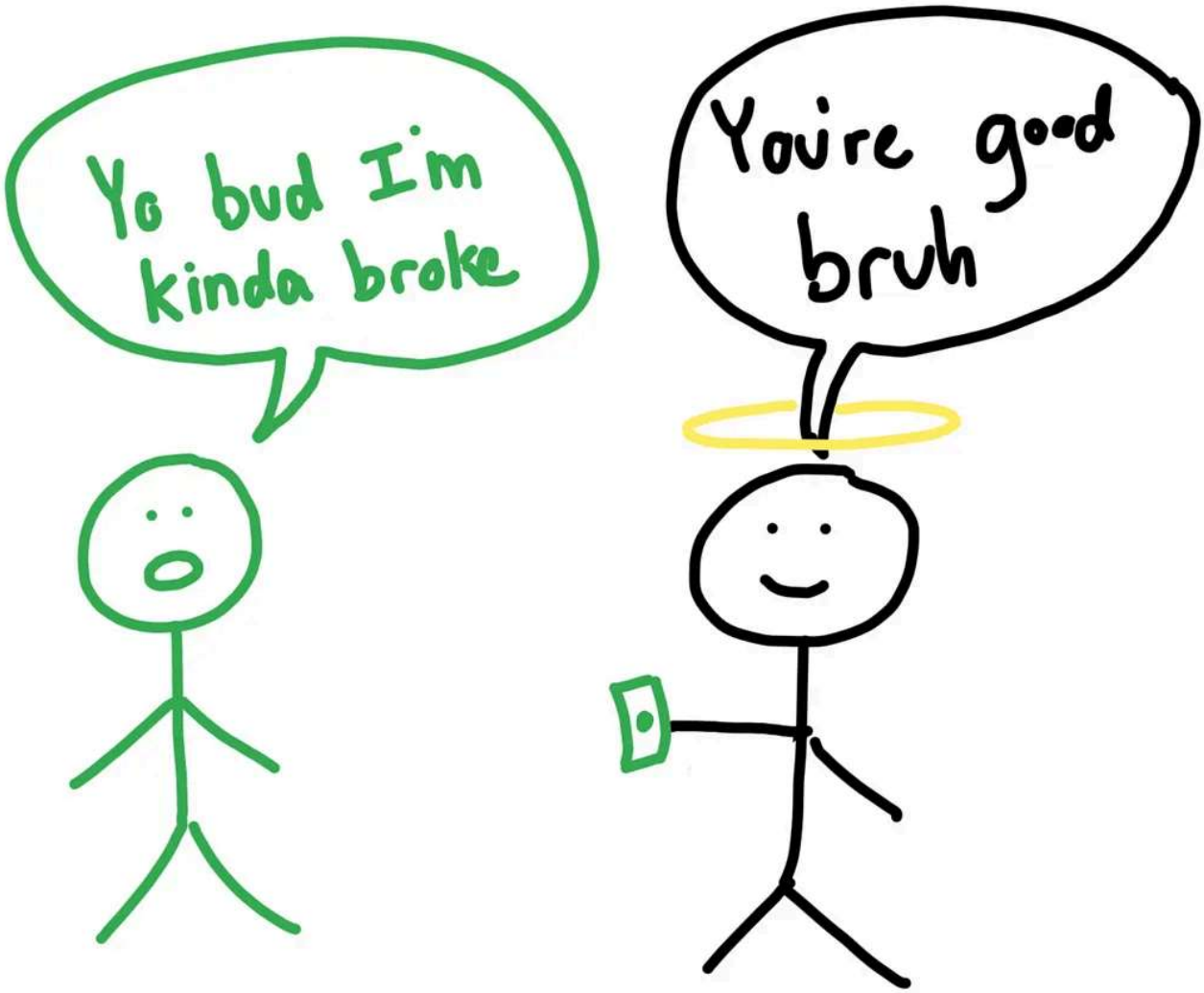

Image created by author

But what if you had hundreds of Teds asking you for money?

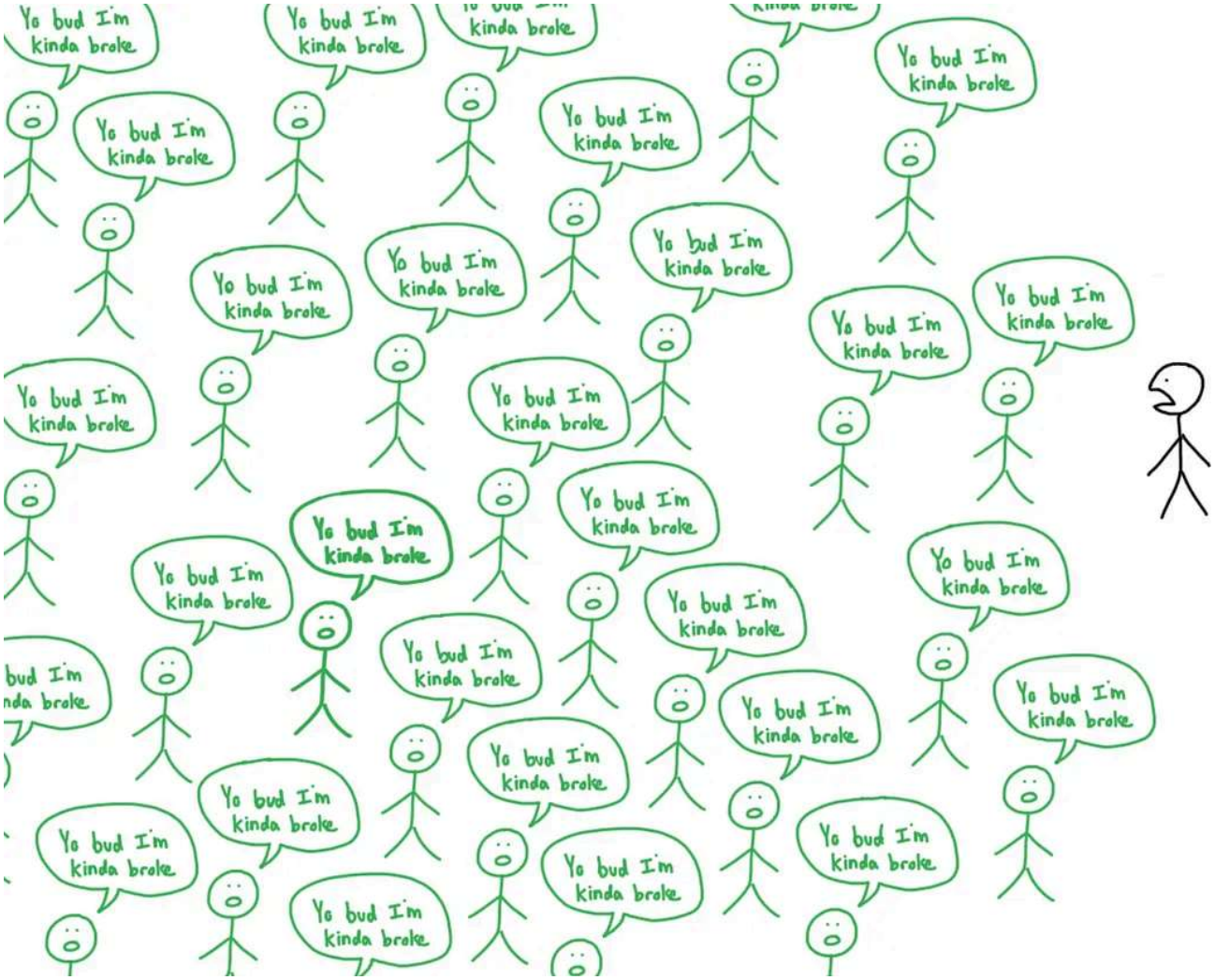

Image created by author

You can't get to know every one of these people so that you can decide whether loaning to them feels like a good idea. But if you just loaned to them randomly, you might loan money to some people who are never going to pay you back. You also might reject some perfectly responsible people who would pay you back quickly. Instead, you'd want to use a more scientific process.

This scientific process is called **credit risk modeling**, and it's what we'll be exploring in this article.

Formally speaking, **credit risk modeling** is the process of using data about a person to determine how likely it is that the person will pay back a loan.

Based on the name of the process, it's no surprise that credit card companies do credit risk modeling all the time. But credit risk modeling doesn't necessarily have anything to do with credit cards, even though "credit" is in the name. Credit risk modeling applies to *any* loans, not just loans associated with credit cards.

Countless organizations use credit risk modeling, including insurance companies, banks, investment firms, and government treasuries. Sometimes, individual people make a *living* using credit risk modeling to strategically loan away their own money. Credit risk modeling is ultra important anywhere people are borrowing money.

Even more importantly, understanding the process of credit risk modeling will make it easier to understand *any* type of modeling that involves probabilities.

But for now, let's come back to Ted so that we can get a solid grasp of the fundamentals of loans.

What are interest rates?

If you're lending money to Ted just to be nice, you'll probably just ask him to give you back the same amount of money you gave him. But you could also take this opportunity to make a bit of extra money.

You could tell Ted that he has to give you back the same amount of money you gave him, *plus* ten percent.

This means if you give Ted $10, he'll have to give you $11 back. The extra money he gives you back is called the **interest**. In this case, Ted is paying you

$1 in interest, and the **interest rate** is ten percent. If you raised the interest rate to 20%, Ted would have to pay you $2 in interest, which means he'd have to pay you $12 in total.

If Ted really needs the money, then he'll probably be okay with paying some interest. *Paying interest on a loan is basically buying the ability to spend money that you'll have in the future.*

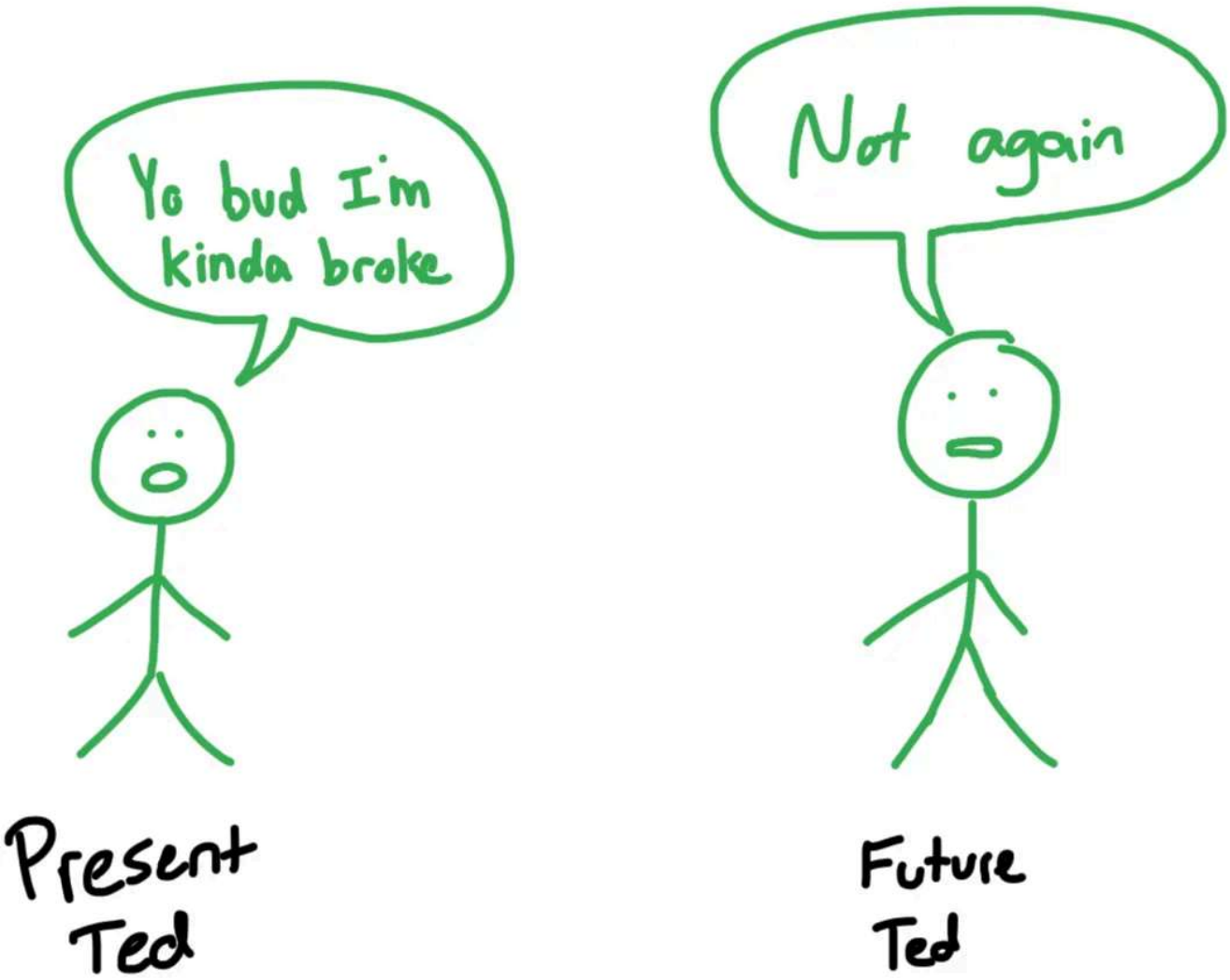

Image created by author

Spending future money in the present is a convenient thing to do, and like all convenient things in the business world, it comes with a price.

With interest rates, you can see how it could be pretty useful to have 500 Teds asking you for money at the same time. If you charged all of them 20% interest, then loaning $10 to each of them could make you a thousand dollars.

But we get to that figure of $1000 by making a pretty big–and invalid–assumption: that everyone will pay you back.

If somebody doesn't pay you back, then you permanently lose the $10 you gave them. You also miss out on the $2 they would have paid you in interest. This means that for each person who doesn't pay you back, you miss out on $12.

If just 84 out of the 500 people don't pay you back, then you'll actually *lose money* from this whole lending process. It's amazing that it takes less than 17% of the people to screw this whole system up.

To make a living off of this, you need a better system.

What is default risk?

Fortunately, credit risk modeling lets you estimate the probability that each person will fail to pay you back. **Defaulting** on a loan means failing to pay it back, so each person's probability that they'll fail to pay back their loan is called their **default risk**.

Determining somebody's default risk is important. Once you know somebody's default risk, there's a way to calibrate their interest rate to mitigate the risk of lending money to them.

But before we can understand how to use a default risk to calibrate somebody's interest rate, it's important to understand a fundamental statistics concept: the Law of Large Numbers.

The Law of Large Numbers

Let's start with a simple situation: flipping a coin.

We've all heard that there's a 50% chance of getting a heads and a 50% chance of getting a tails. This is called a **theoretical probability**.

If you flip the coin once, then you're going to get either heads or tails. If you flip the coin 10 times, you could get 5 heads and 5 tails (50% heads). But it's also totally possible that you get 6 heads and 4 tails (60% heads), or even 9 heads and 1 tails (90% heads).

These observed results are called the **experimental probability**. The experimental probability doesn't necessarily equal the theoretical probability.

So what does a 50% chance of getting heads actually mean?

It means that if you flip a coin a lots and lots and lots of times, the overall result will be that about 50% of the flips come of up heads.

Let's look at this step by step.

Let's flip a coin 5 times. The number of heads we got is the number of red dots, and the number of tails we got is the number of blue dots.

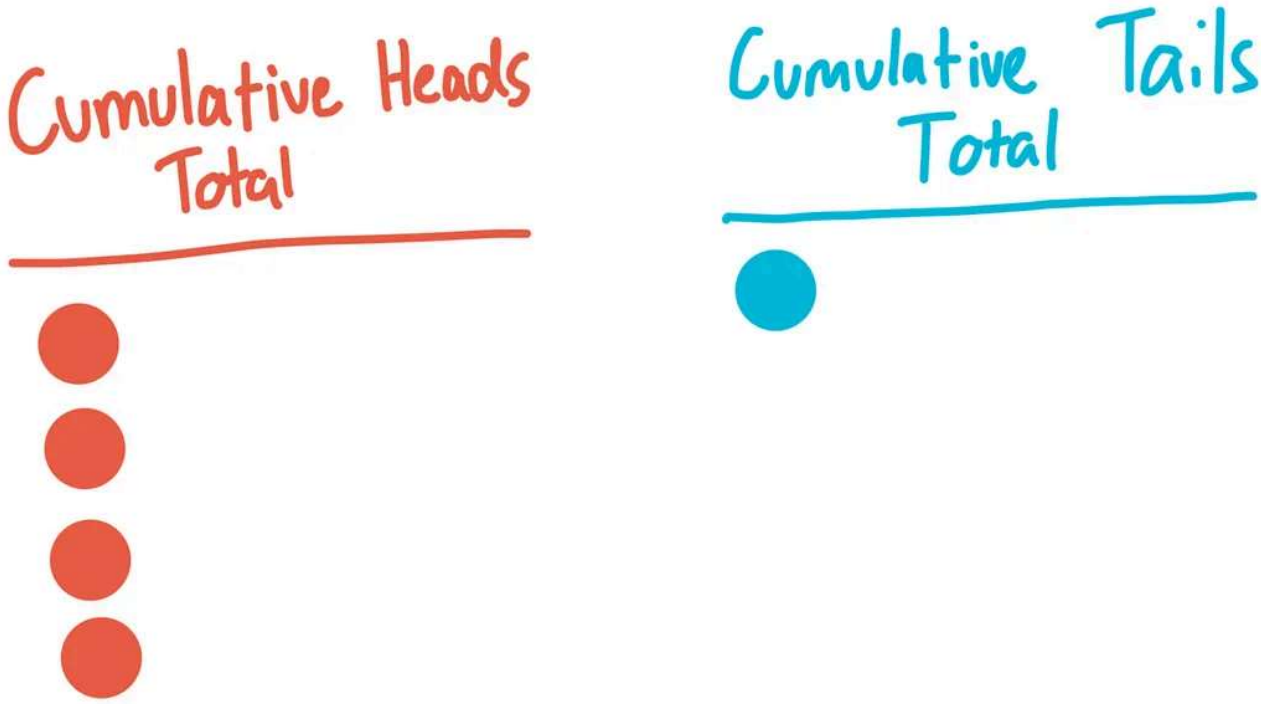

Image created by author

This looks pretty imbalanced so far, with 80% of our flips coming up heads. That's an 80% experimental probability, which is pretty far from our 50% theoretical probability. But let's keep flipping. Here's what we've gotten after flipping the coin 15 times.

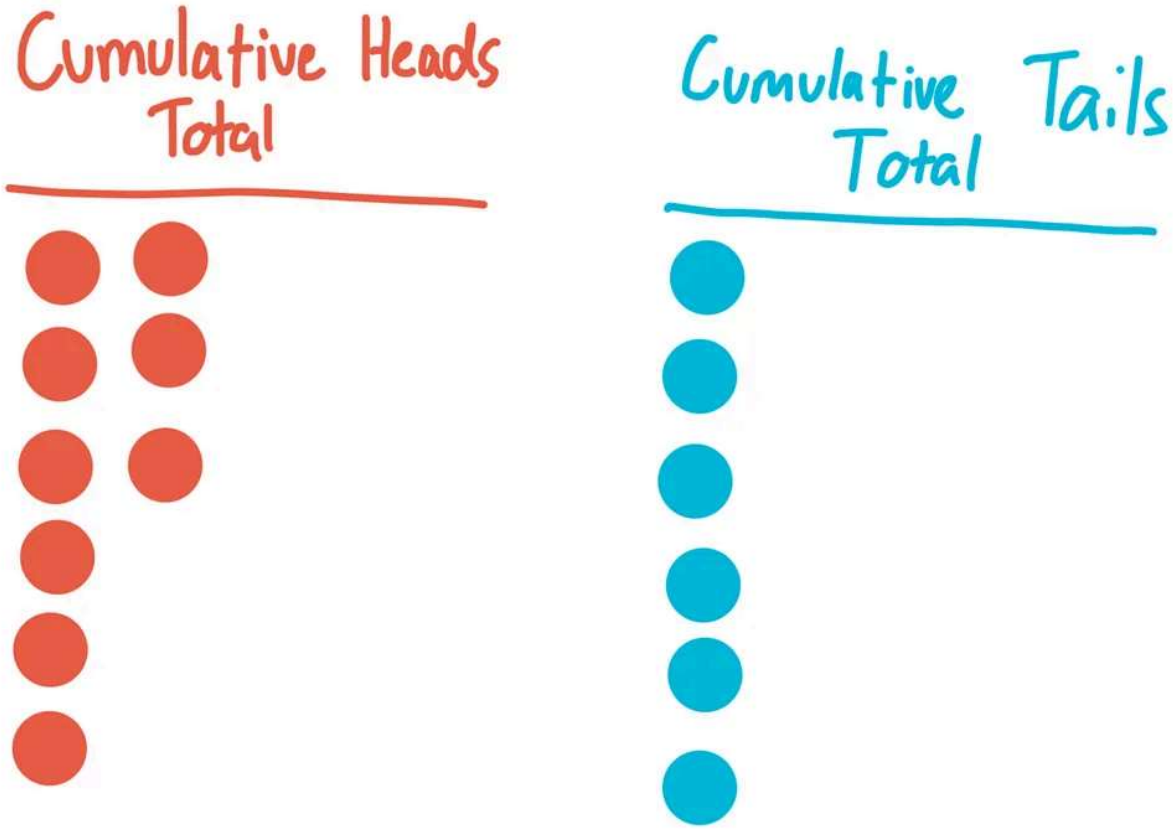

Image created by author

Now we've had a total of 60% of our results come up as heads, which is a lot closer to 50%. If we keep flipping until we've flipped the coin 50 times, here's what we get.

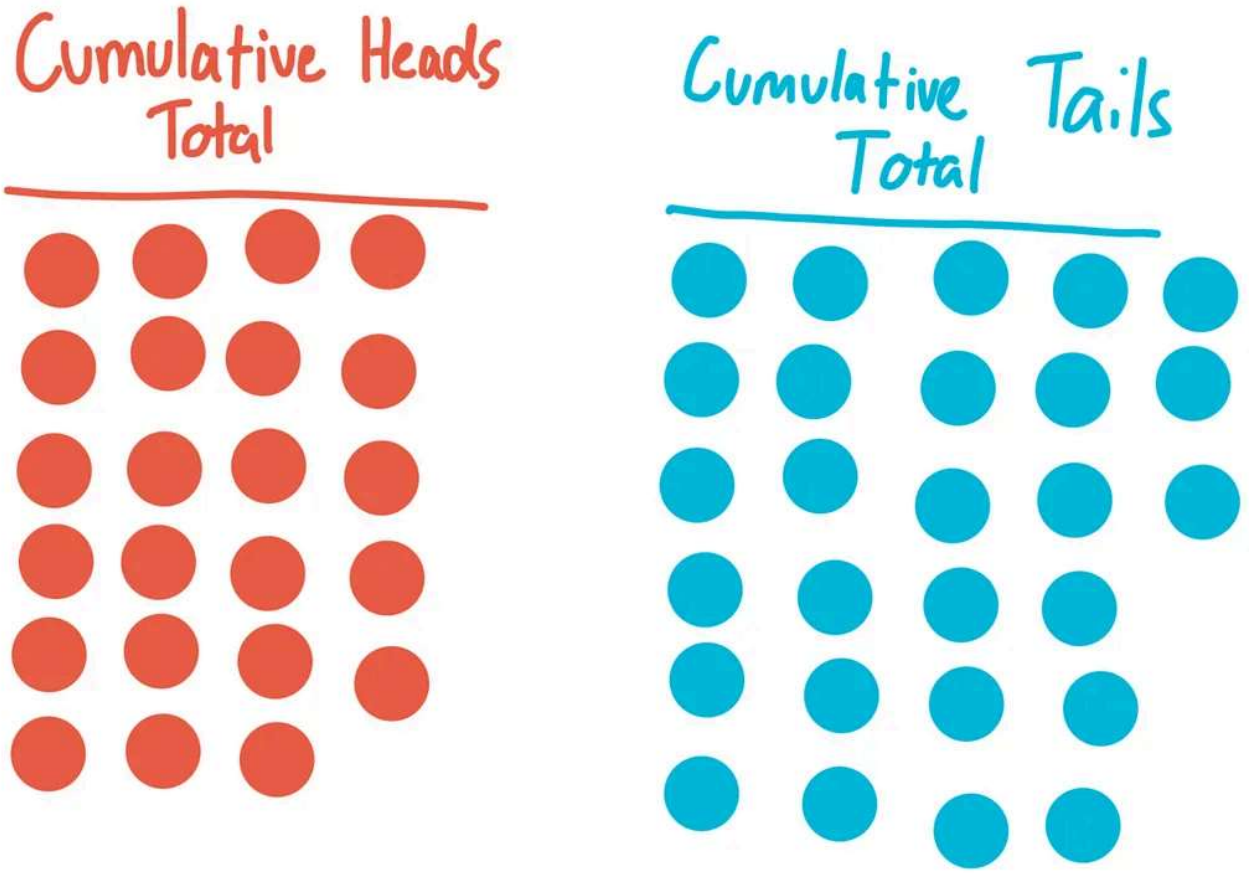

Image created by author

That's 46% of our total flips now coming up heads, which is even closer to 50%. If we keep flipping the coin over and over again, then we're going to see this percentage of heads get closer and closer to the theoretical probability of 50%. It might not get closer to 50% with every single flip, but it will generally tend to get closer to 50%.

If we keep flipping, until we get to 100 flips, the blue line represents the cumulative percentage of heads over time. The red line is 50%.

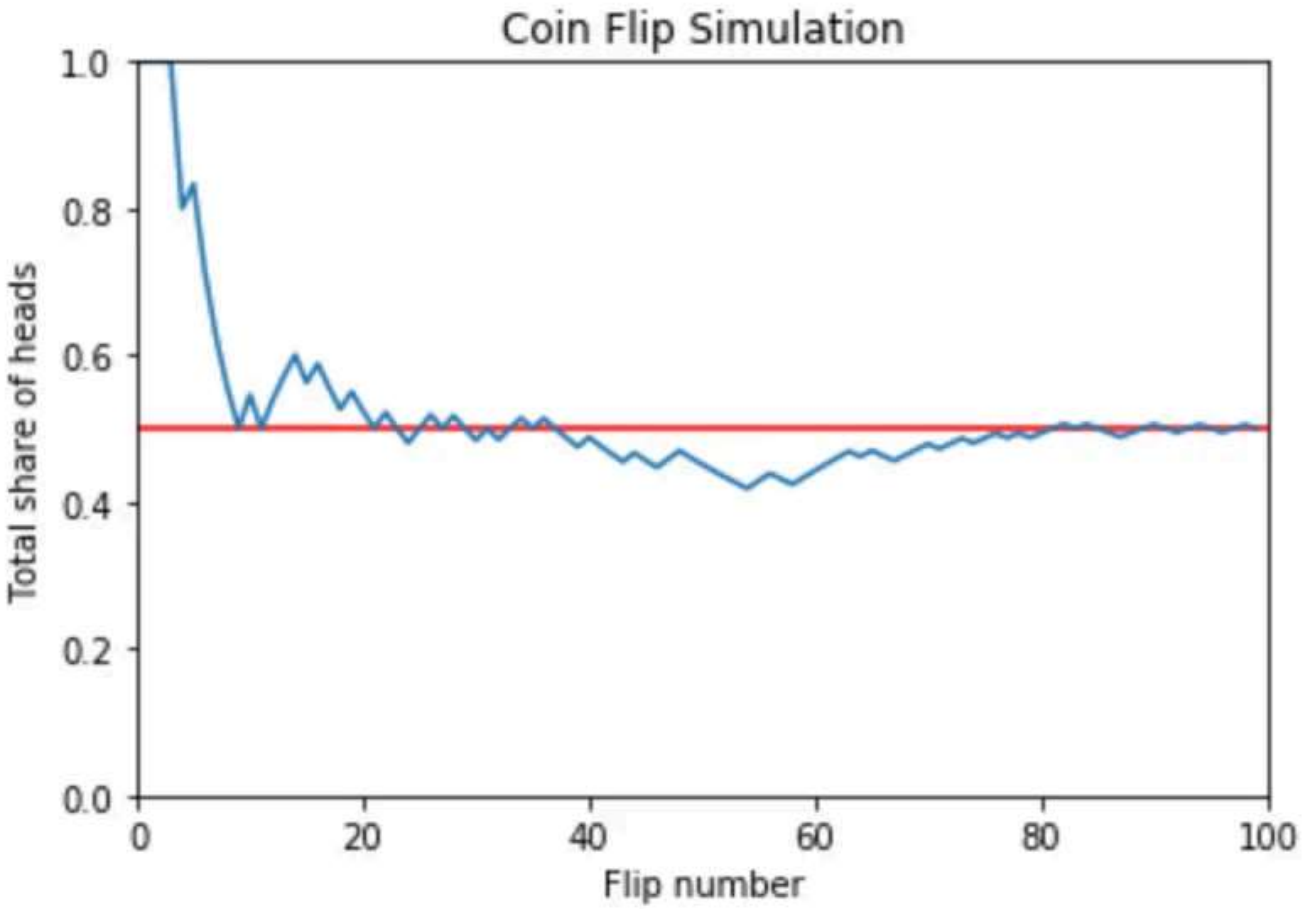

Image created by author

The blue line starts off really high because our first four flips were heads, which means that 100% of our cumulative tosses came out heads for a little bit. But as we get some tails in the mix, the blue line tends to get closer to the red line.

As you can see, as we flip the coin more and more times, the blue line, which represents our experimental probability, approaches the red line, which is our theoretical probability.

This is how we define a theoretical probability, like a 50% chance of getting a heads or a 16.7% chance of getting a 1 when rolling a 6-sided die.

The Law of Large Numbers is that the experimental probability tends to approach the theoretical probability as we do a large number of trials. The more trials we do, the closer the experimental probability tends to get to the theoretical probability.

To illustrate this concept, here's a coin flip simulation graph with fifty thousand flips:

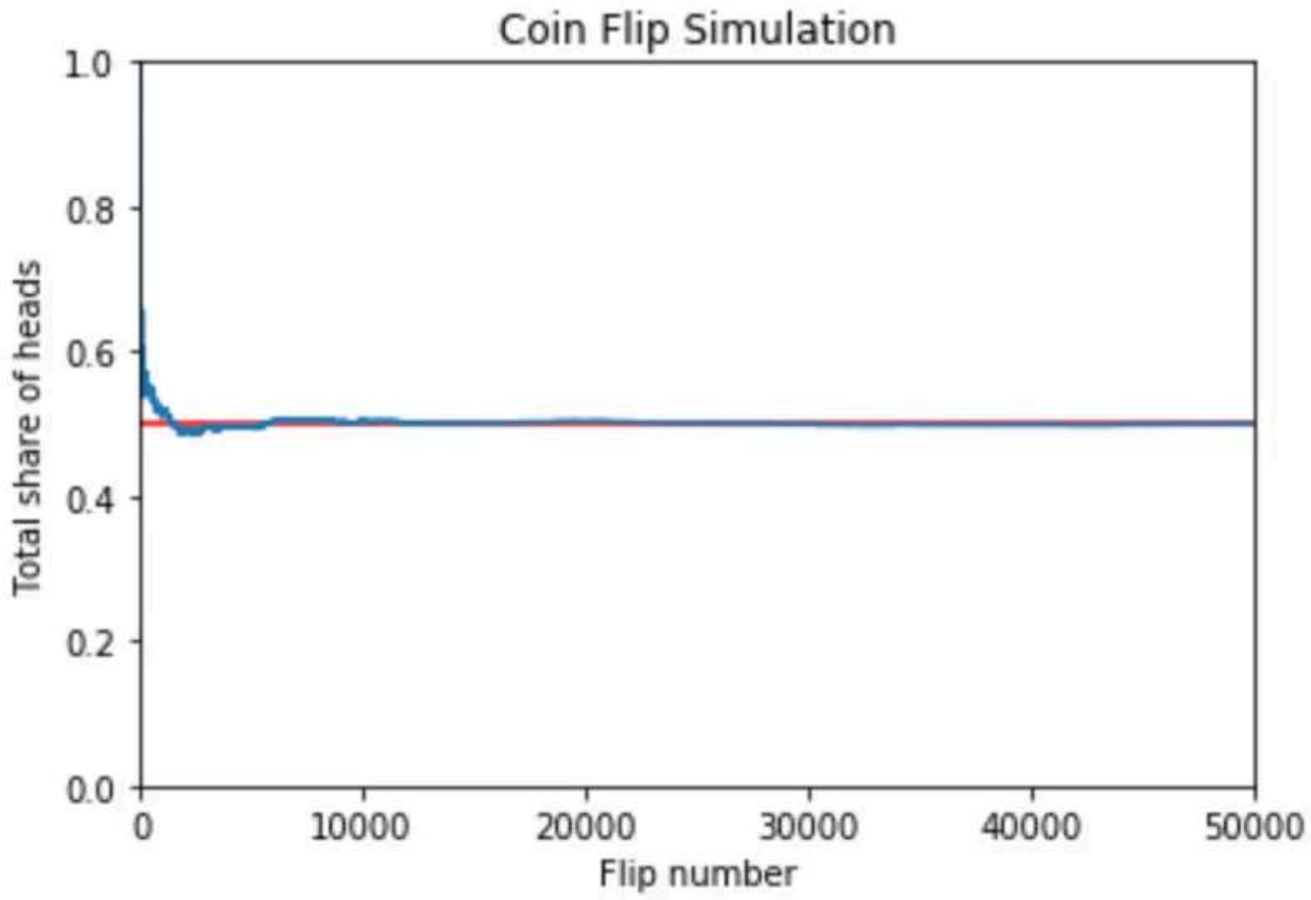

Image created by author

By the end, the experimental probability gets so close to the theoretical probability that the blue and red line are almost on top of each other.

This means that when we're dealing with a ton of trials, we can assume that the experimental probability will basically equal the theoretical probability. In this case, if we flip a coin fifty thousand times, the experimental probability will be super close to 50%.

It's important to highlight that each time you flip the coin, *the probability of getting a heads is always 50%, no matter what came before it*. Even if you've gotten 10 heads in a row, the probability of getting another heads on the next flip is *still* 50%.

This helps explain why the blue line in the graph above doesn't always go straight to the red line. Even near flip 30,000, you can see that the blue line still wiggles just a little. But over time, the blue line generally tends to approach the red line.

The important thing to take away from the Law of Large Numbers is that a large number of trials gets the experimental probability really close to the theoretical probability.

If you think this idea is making a bit of sense, you're ready to understand the process of calibrating interest rates based on default risk.

Using default risk to calibrate interest rates

Let's say you're thinking about making loans of $10 to a large number of people, such as 10,000 people. Using credit risk modeling, you discover that on average, each person's theoretical default risk is 15%. The Law of Large Numbers tells you that the experimental probability approaches the theoretical probability for a large number of trials. Given that you have a

large number of people here (10,000 people), it's safe to assume that about 15% of the people you loan to really will default, and 85% will pay you back.

If you know that about 15% of people probably won't pay you back, then you know that you're going to permanently lose 15% of the $10 loans you make. Because 15% of 10,000 people is 1500 people, and you're loaning each person $10, you know you'll probably lose $15,000.

But you don't need to just be okay with losing $15,000. There are a few things you can do here.

1. Increase everyone's interest rate to make up for that $15,000.

This way, the 85% of people who actually do pay back their loan will end up paying a total of $15,000 extra. Because 85% of 10,000 people is 8,500 people, that's about an extra $1.76 of interest per person. In this case, that's equal to raising the interest rate by 17.6 percentage points.

Here's the issue: if the interest rate is already set at 20% so that you can make some profit, then raising the interest rate another 17.6 percentage points makes the total interest rate 37.6%. That's a big increase, and increasing the interest rate like that might deter some potential borrowers.

2. Only accept the safest borrowers.

Credit risk modeling lets you see each individual's default risk separately. Some people might have a 40% default risk, while others might have just a 1% default risk. If you only give loans to people whose default risk is below a certain threshold, say 2%, then the percentage of people who will pay you back will be much higher. This means that fewer people will default, and you'll lose less money.

As a result, you can charge a lower interest rate, because you won't have to make up for as much lost money due to defaulting. The problem with this strategy is that you might have to turn away a big portion of people who want a loan, which would decrease your profits.

3. Charge people interest rates that are proportional to their default risk.

This is the most important strategy in the real world. As we go through all the borrowers, we'll call each individual borrower "borrower x". **For each borrower x, you charge borrower x an interest rate such that you wouldn't lose any money if every borrower had borrower x's default risk.**

That's a dense idea, so let's unpack it.

When analyzing the first strategy, we discovered that we should add 17.6 percentage points to each borrower's interest rate if each borrower has a 15% default risk. (This is the same as 15% being the average default risk.) This way, the extra interest paid by the 85% who don't default will make up for the 15% of people who default.

If everybody you're loaning money to has the same default risk, then there's always a certain amount that should be added to each borrower's interest rate to offset the people who default.

Default Risk (Percent chance of defaulting)	Extra Interest (In percentage points)
1%	1.01%
5%	5.26%
10%	11.1%
15%	17.6%
20%	25.0%
40%	66.7%

Image created by author

In fact, the amount that should be added to each borrower's interest rate can be calculated with the following formula:

$$\text{Extra Interest} = \frac{\text{risk}}{1 - \text{risk}}$$

Image created by author

There's no need to delve into this formula if you don't want to, but if you want to learn how it works, this paragraph will show you how. The risk term on top represents the share of people who are going to default. Defaulting is a binary outcome; this means that every person is either going to default or not default. There's no in-between. The number 1 represents 100% of the people, so the term 1 — risk (one minus risk) is the share of people who don't default. You divide the defaulters by the non-defaulters to determine what percentage of each defaulter's loan the non-defaulters need to pay off.

Even if each borrower's default risk is different, you can use this formula to calculate exactly how much you should add to each borrower's interest rate to counterbalance their risk. If you add this amount to every borrower's interest rate, and you have a large number of borrowers, the Law of Large Numbers shows us that you almost certainly won't lose money in the long run.

Just to recap, here's a breakdown of the money somebody might pay you back after you loan money to them.

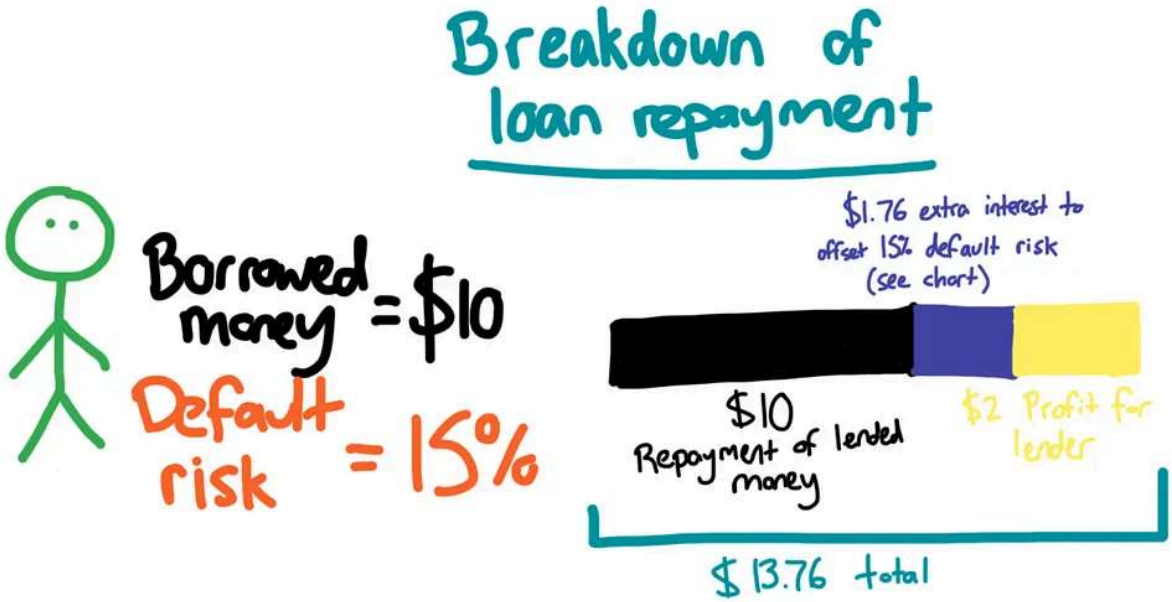

Image created by author

In the real world, these strategies are combined. In accordance with strategy 1, lenders often increase everyone's interest rate by small fixed rate, regardless of default risk, to make up for uncertainty in their models. In accordance with strategy 2, lenders often have a risk threshold, so they won't accept everyone. And finally, in accordance with strategy 3, lenders use each borrower's default risk to add a certain number of percentage points to each individual's interest rate.

If you're getting the hang of these three strategies, then you have a solid grasp of the fundamentals of loaning money.

At this point, we've explored why finding somebody's default risk is so useful. Now, we're ready to dive into the machine learning models used in credit risk modeling to calculate these default risks.

What is machine learning?

People toss around the term "machine learning" a lot nowadays, and sometimes it's hard to know what it means.

Simply put, a machine learning model is a mathematical predictive model that gets better as it sees more data.

To do machine learning, you need two things: a model, and data.

There are tons of different types of machine learning models. A machine learning model is the name for the set of steps that are used to make

predictions based off the data. Below, we'll explore four fundamental machine learning models that are important in credit risk modeling.

You also need historical data, and lots of it. This data needs to contain both the thing you're trying to predict (called the **target**) and other characteristics that are related to the thing you're trying to predict (called the **features**). In the case of credit risk modeling, you need historical data for lots of individual people. The target data here is whether or not each person defaulted. The feature data here consists of statistics about each person, such as their income, age, and employment status.

Once you have your data, you need to **train** the model by feeding it historical data. Machine learning models are built to connect the feature data to the target data. After training, the model can look at cases where the feature data is known and the target data isn't, and the model can make predictions.

This situation is common in real life. When you're trying to decide whether or not to give someone a loan, you usually have all their feature data, such as income and age. You just don't know the target data: the probability that they'll default on their loan.

Calculating this probability is where machine learning models shine.

Let's start with a simple, but surprisingly powerful model.

The K-Nearest Neighbors Model

Let's say we have a bunch of historical lending data with two features: each person's income and their age. We also know whether each person defaulted

on their loan. If they defaulted, then their dot is colored red on the graph. If they paid back their loan, their dot is colored blue on the graph.

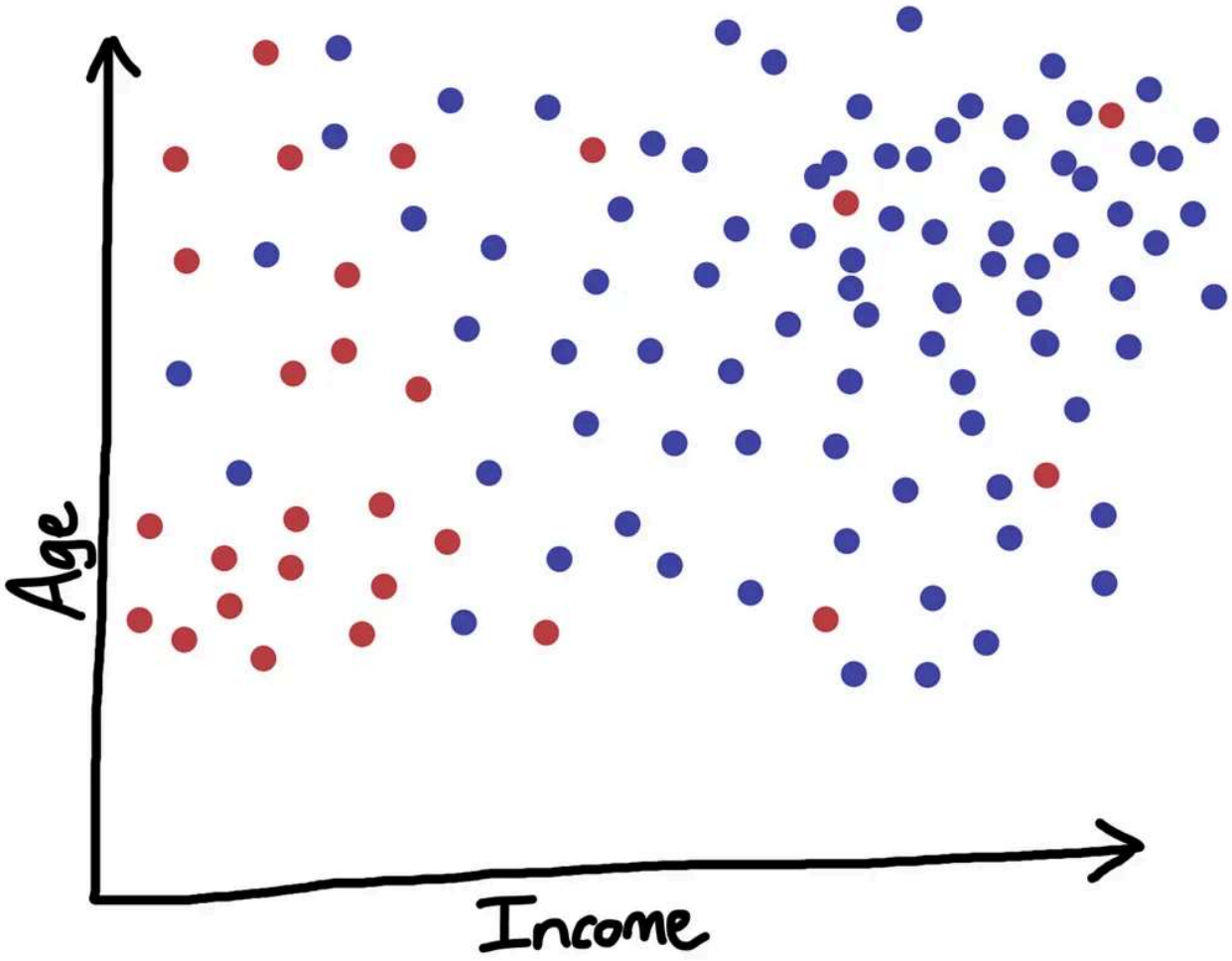

Image created by author

Now let's say Ted is asking again for a loan. We want to know how likely it is that he'll pay back his loan, and we now have a much more powerful tool than our feelings alone. If we know Ted's income and age, then we can plot him in green on the same graph.

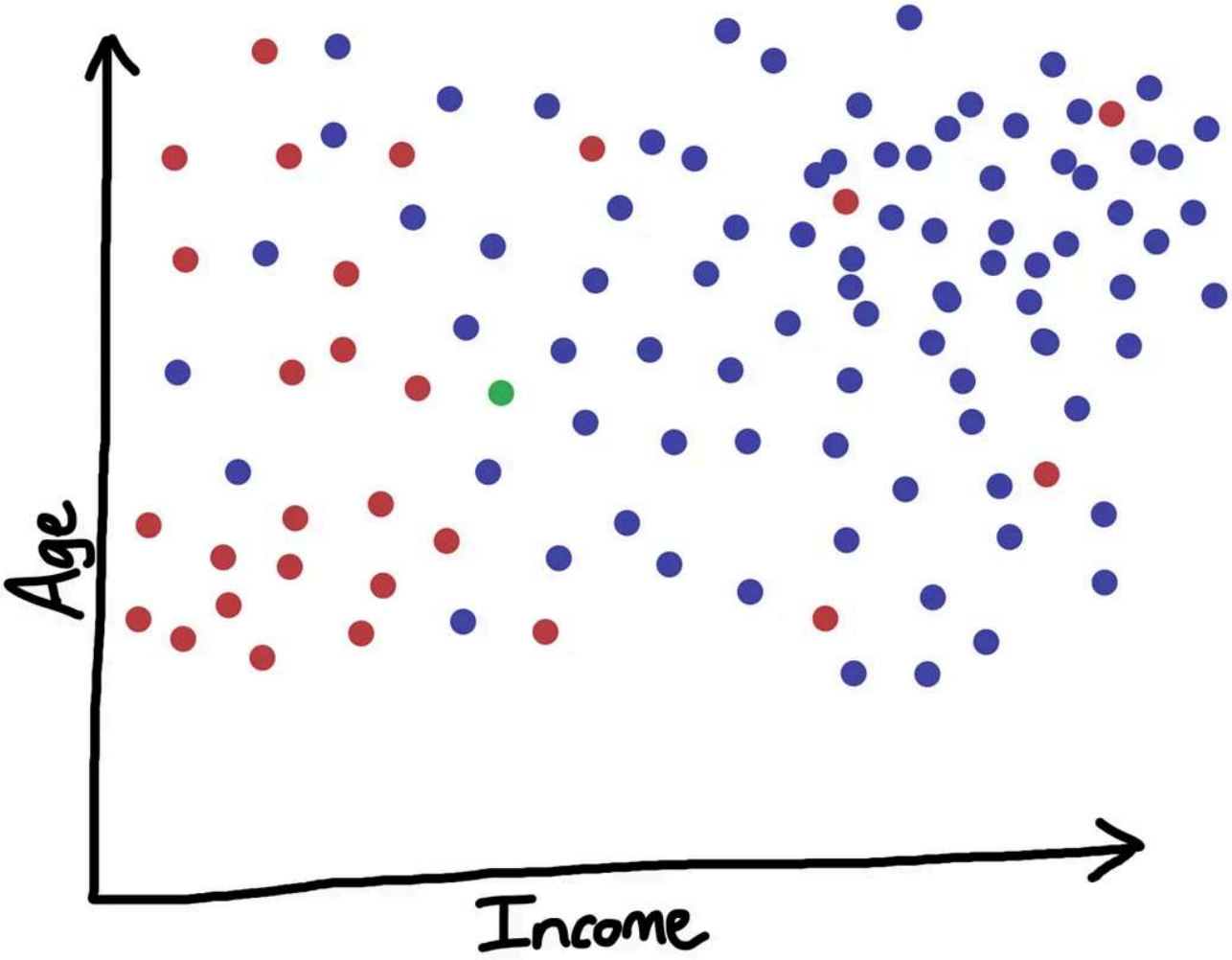

Image created by author

We then select a number of “nearest neighbors” that we’ll look at. This number is called K. If K = 5, then we’ll select the 5 dots that are closest to Ted’s dot and figure out what percentage of them defaulted.

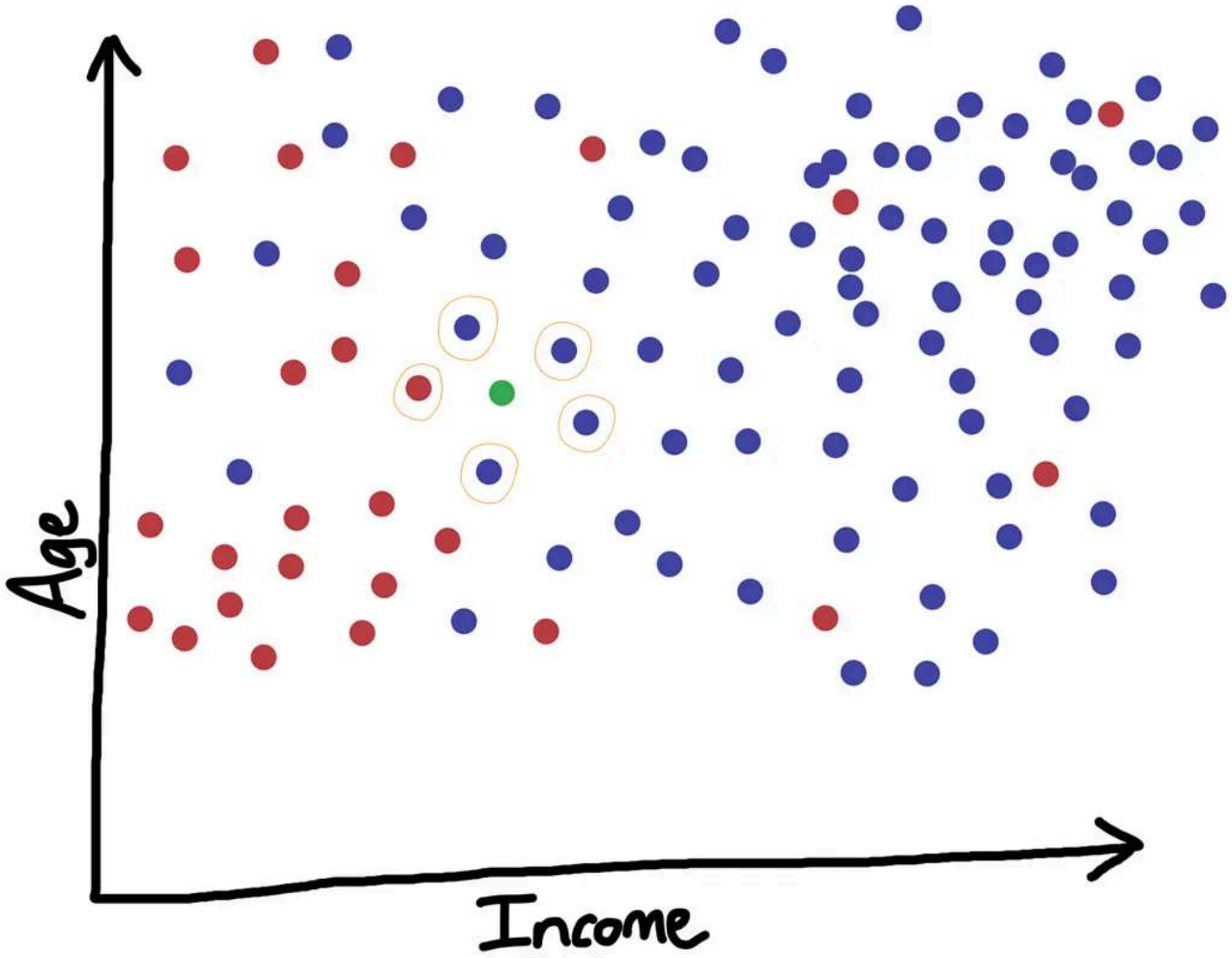

Image created by author

In this case, 80% of people nearest to Ted paid back their loans (4 people), and 20% of them defaulted (1 person). This means that we tentatively estimate Ted's default risk as 20%.

Of course, you're probably thinking that setting K equal to 5 seems pretty arbitrary. There is a way to optimize our K value, and it lies in **testing** the model.

To test the model, we split our historical data into two parts: the **training data** and the **testing data**. We then set our K value to an arbitrary value, say 10, and we feed the model our training data to generate a graph like the one

above. Then, for each person in the testing data, we make our trained model guess whether or not the person defaulted on their loan. The model doesn't get to see the target column of the testing data; it only sees the features. During testing, the model has to make predictions based on the features alone, just like it would in real life.

Each time the model makes a prediction, we record whether the model predicted correctly. We then calculate the model's **accuracy score** by finding the percentage of cases it predicted correctly.

To optimize our K value, we test the model over and over again with different K values until we find one that works best. Sometimes the ideal K value is 1, while other times the ideal K value is huge. (Of course, for credit risk modeling, a K value of 1 would be silly, because this would either give us a 0% or 100% default risk.)

In the 2D graph above, we only had 2 feature variables: income and age. But the k-nearest neighbors algorithm works great with more than 2 feature variables as well. If we had 3 feature variables, we could plot the points in a 3D box instead of a 2D square.

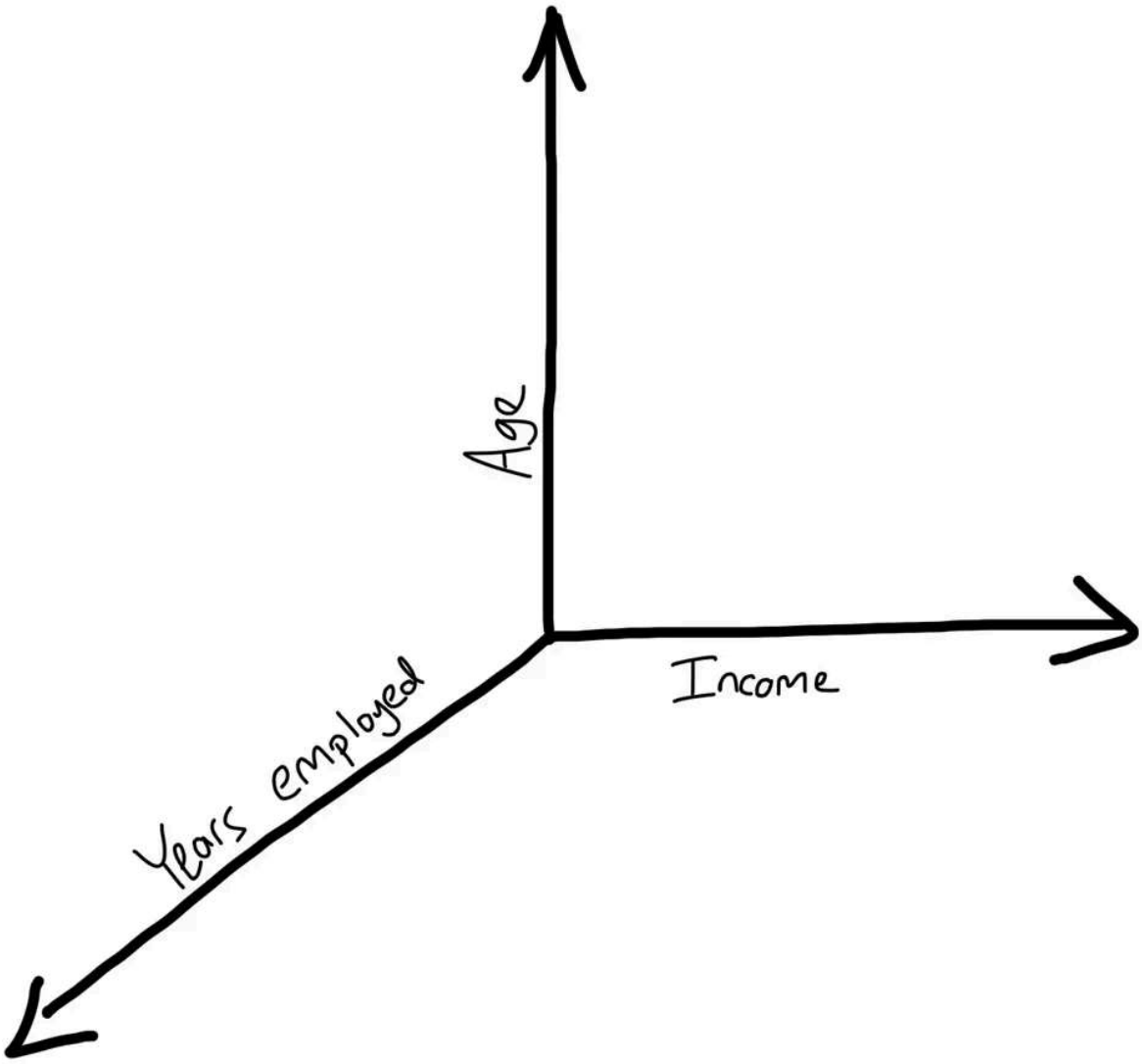

Image created by author

With 3 variables, the same set of steps applies. Training the model gives us a 3D graph with a bunch of blue and red dots on it. When we have a person whose default risk we're trying to predict, we plot them on the graph. To calculate their default risk, we find this person's K nearest neighbors, and we calculate the percentage of these neighbors who defaulted. We say that this percentage is the new person's default risk.

Once we get to 4 variables, the graph becomes hard to visualize. But the computer can deal with a nearly infinite number of variables. In fact, it's not too difficult to algebraically calculate nearest neighbors by hand with 4 or more variables using the Pythagorean Theorem. (Google it if you're interested!)

k-nearest neighbors might seem like quite a simplistic model, but with thousands of data points and many features, it can be quite powerful.

In some cases, however, other models are more effective. Let's discuss another classic machine learning model: logistic regression.

The Logistic Regression Model

The simplest form of logistic regression involves a dataset with a target column and a single feature column. For instance, a historical dataset that contains borrowers' incomes and whether or not they defaulted on their loan would be very simple to use for logistic regression.

For logistic regression, the target data must be **binary**, which means that it can only have two outcomes. In credit risk modeling, the target data is indeed binary: a person can either default or not default on a loan.

In order to make a graph, we can represent this binary outcome as the probability that *a person in the past* paid back their loan. If somebody defaulted on a loan in the past, then we can say that there was a 0 probability that they paid back their loan. If somebody actually did pay back their loan, then we can say that the probability that they paid back their loan was 1. It might feel weird to assign probabilities to past events, but probabilities of 0%

or 100% are just another way of saying that we know something either definitely didn't happen or definitely did happen.

Here's what a graph of income vs historical default probability might look like:

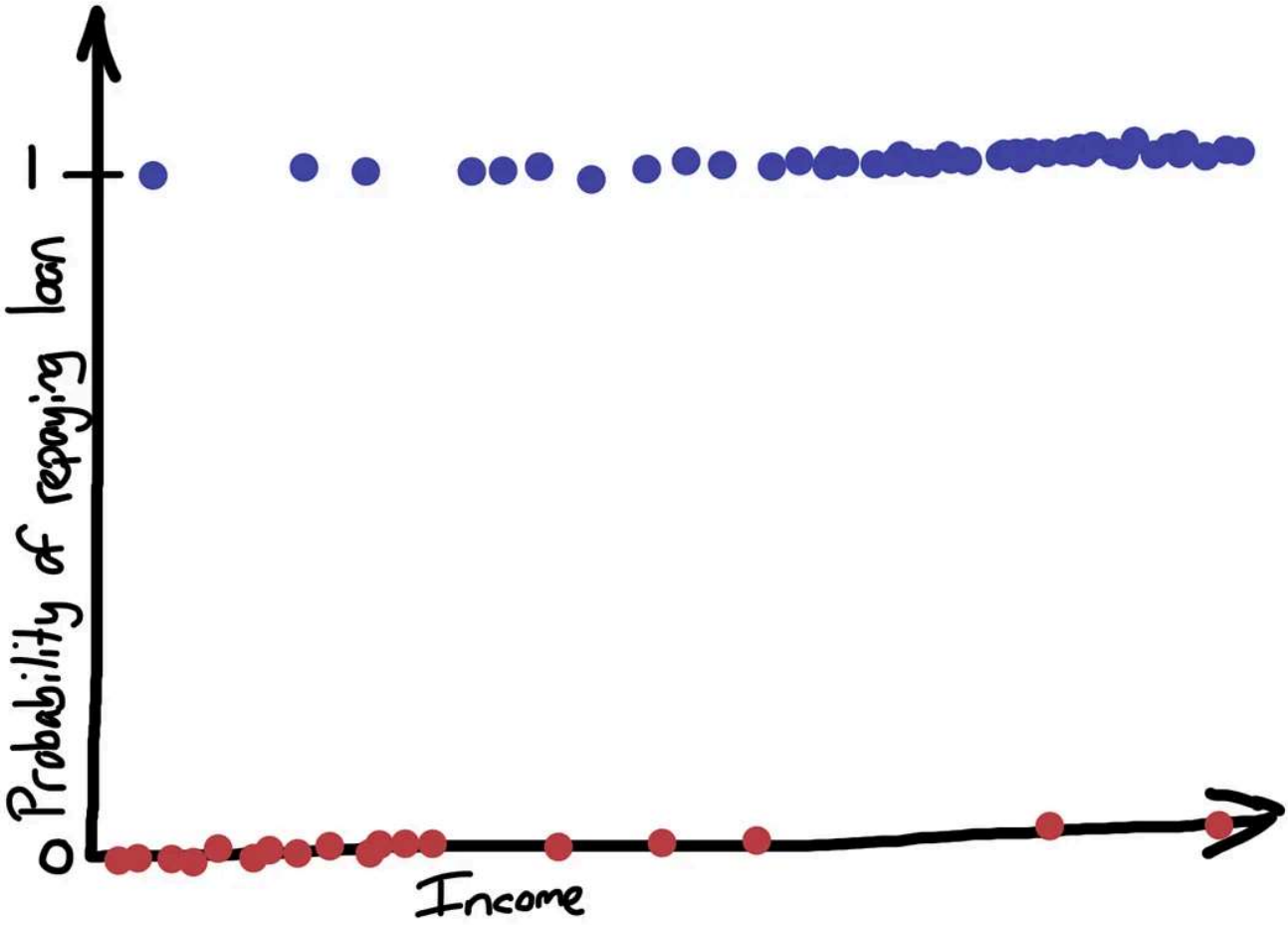

Image created by author

Because we're dealing with definite past data, all the dots are either on the top, or bottom. A blue dot on the top represents a person who did pay back their loan, while a red dot on the bottom represents a person who defaulted.

This is where things get cool. Using some interesting math that we won't dive into here, the logistic regression algorithm calculates a line that looks

something like this:

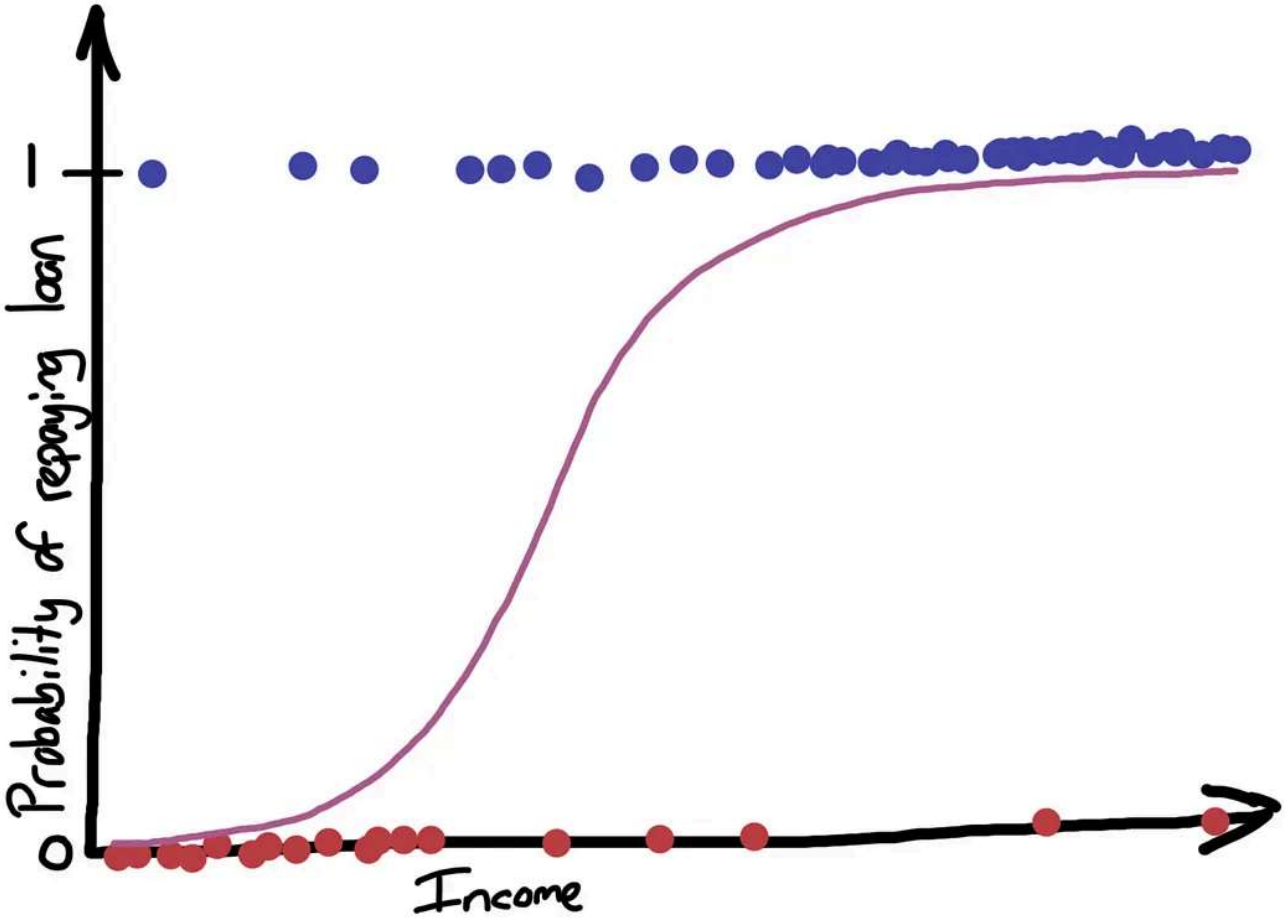

Image created by author

This line tells us the probability that a new person with a given income level will pay back their loan.

Predicting a new person's default risk with logistic regression is pretty simple. You just find the point on the income axis that corresponds to the new person's income, and the logistic regression line's y-value that corresponds to this point gives you the probability that they'll pay back the loan. Subtracting this value from 1 gives you their default risk.

Let's say that Ted's income is right here:

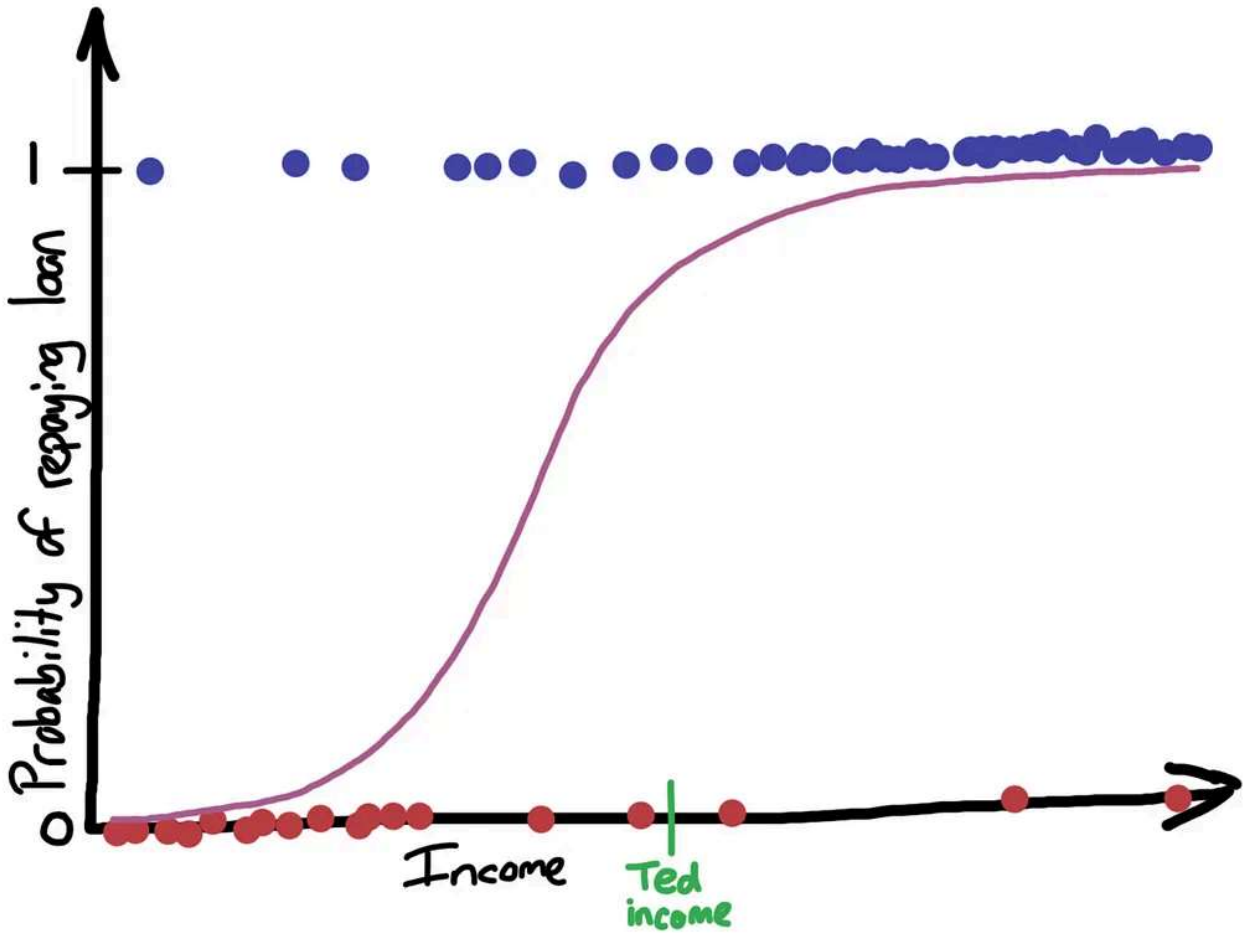

Image created by author

To find the probability that Ted will pay back his loan, we go up and meet the logistic regression trend line.

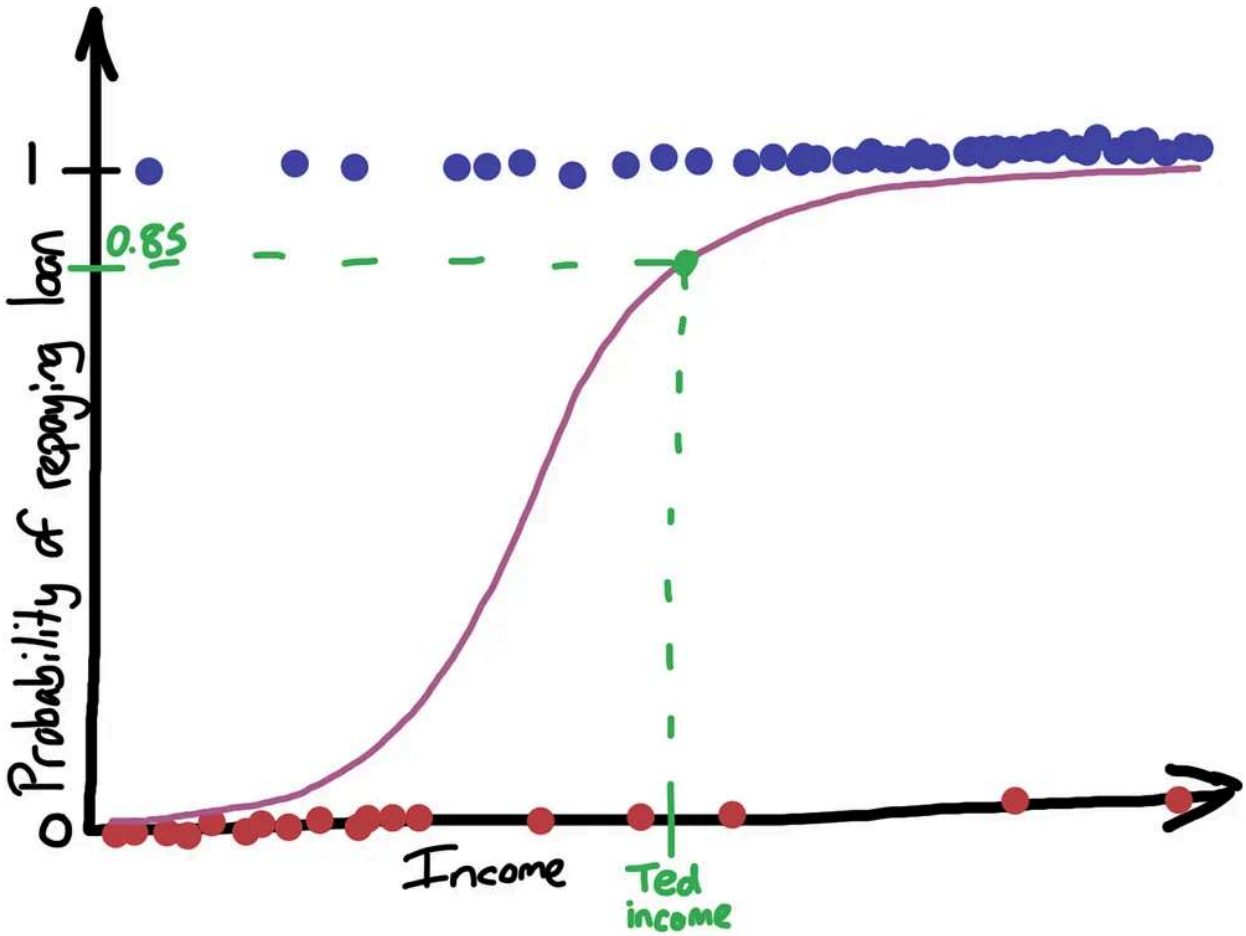

Image created by author

Based on this model, it looks like he has about a 85% chance of paying back his loan. This means that his default risk is 15%.

It's worth noting that this version of logistic regression put Ted's default risk at 15%, while our k-nearest neighbors model put Ted's default risk at 20%. Getting different predictions from different models is totally normal in the world of machine learning. To decide which model to use, we would have to do a bunch of train/test cycles with historical data to see which model consistently performs best in this scenario.

A 2D version of the logistic regression model is pretty simplistic, just like 2D k-nearest neighbors. But just like K-Nearest-Neighbors, a computer (or a

person with a pencil!) can extend logistic regression to an infinite number of dimensions without any problem.

K-nearest neighbors and logistic regression are both great models for making approximations, especially if the data follows a linear pattern. But sometimes, it's nice to have a model that picks up on subtle, non-linear patterns in the data. The decision tree model is a simple model that's excellent at finding such patterns.

Decision Trees

To understand decision trees, let's go back to the graph we used to illustrate how k-nearest neighbors works.

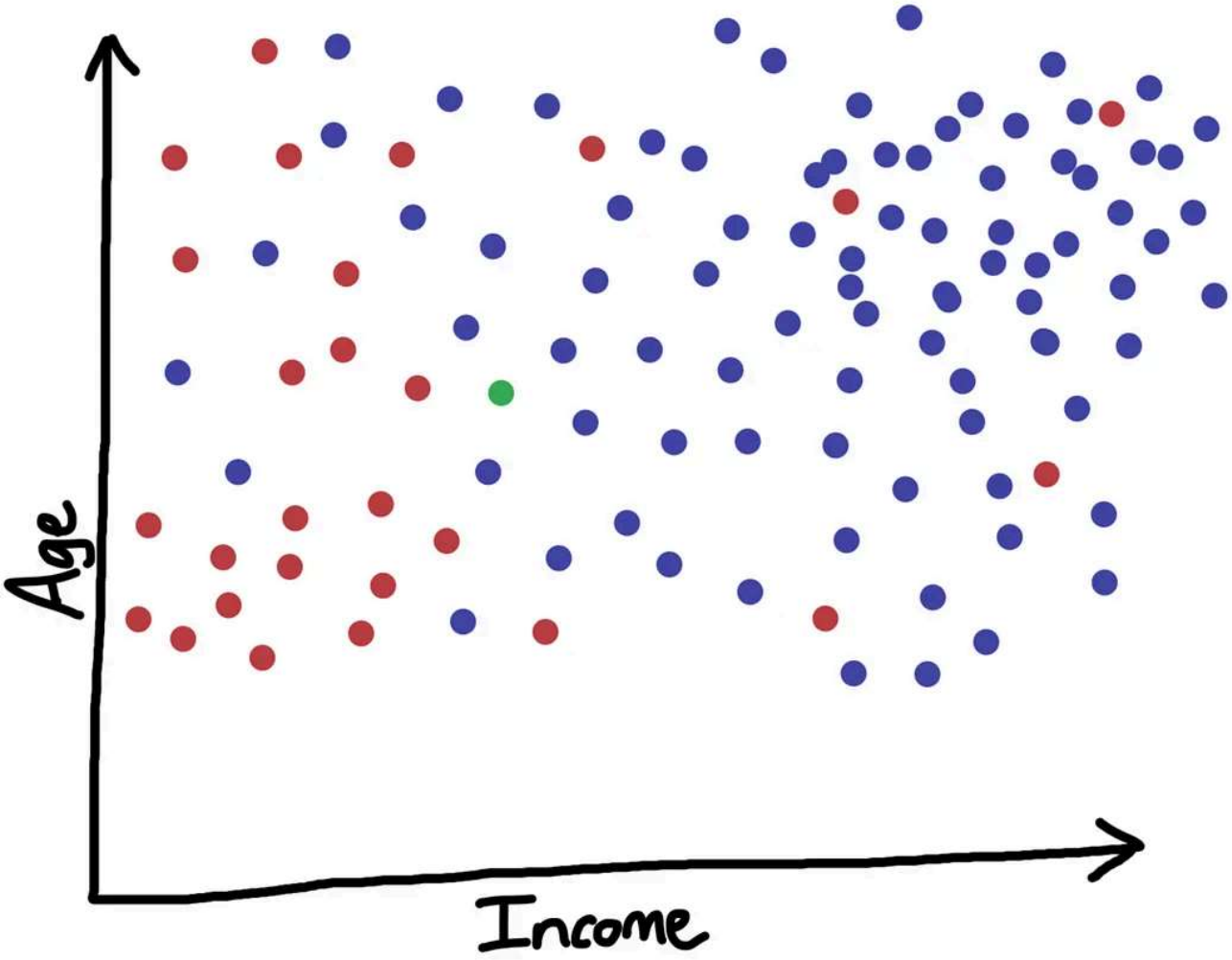

Image created by author

As we've seen, one way to make sense of this graph would be to use the k-nearest neighbors algorithm. However, we could also make some simple rules for how we classify points based on their location. For instance, we could draw a line that splits the graph in half, like this:

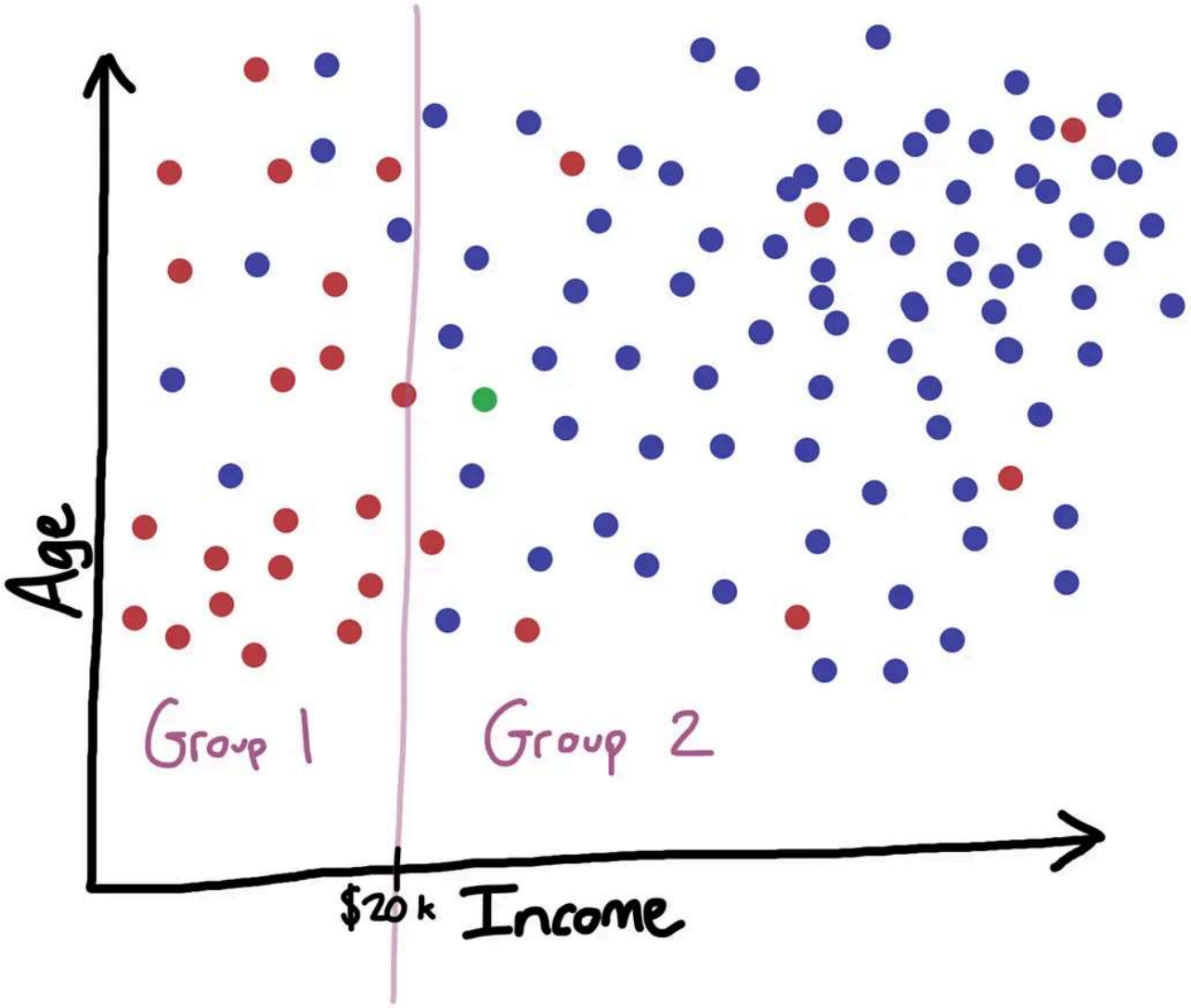

Image created by author

Drawing this line splits the data into two halves. One way to represent this is using a tree diagram, as we can see below.

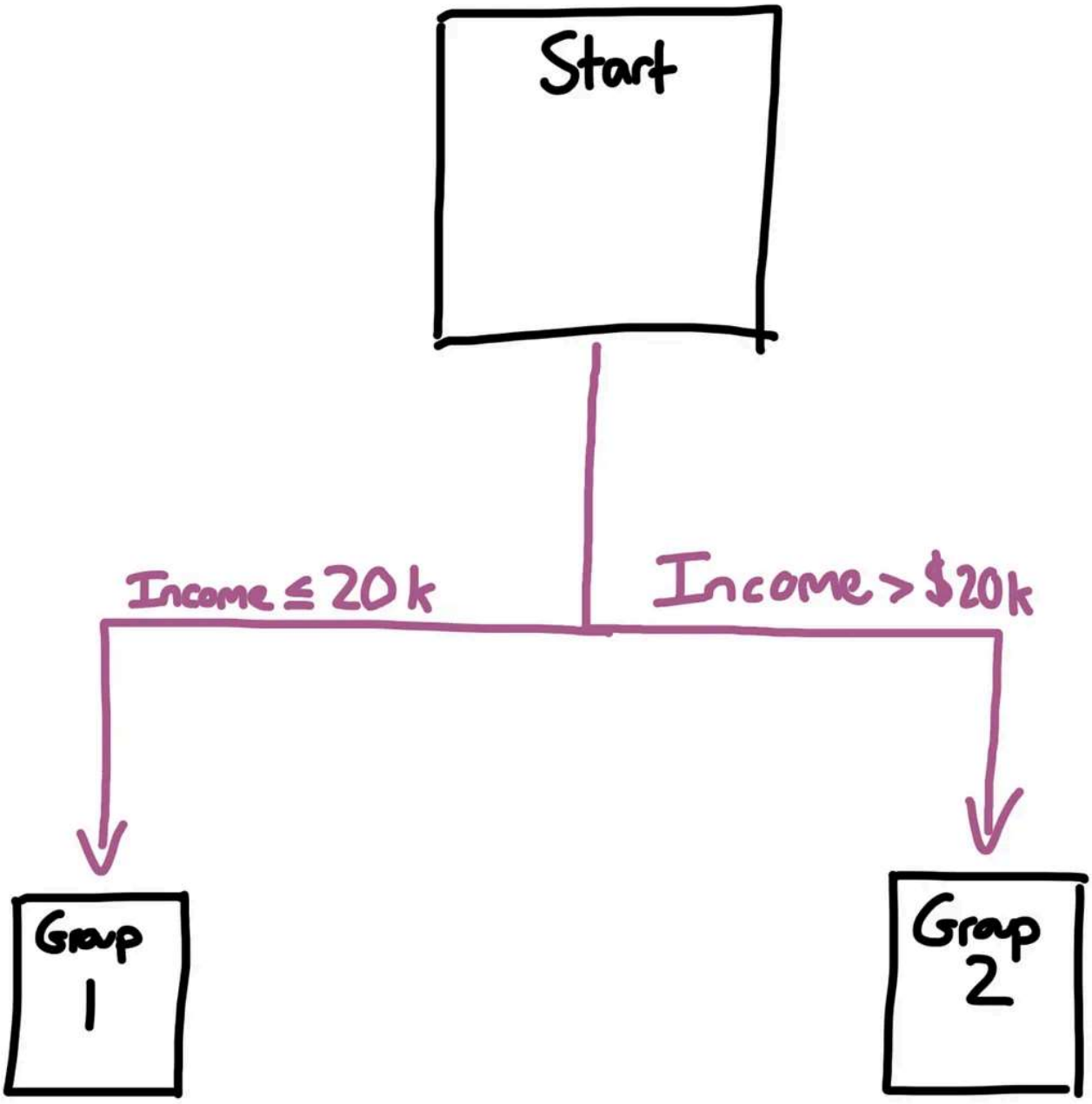

Image created by author

Then, we can split each of these splits again, so that they end up looking like this on the graph:

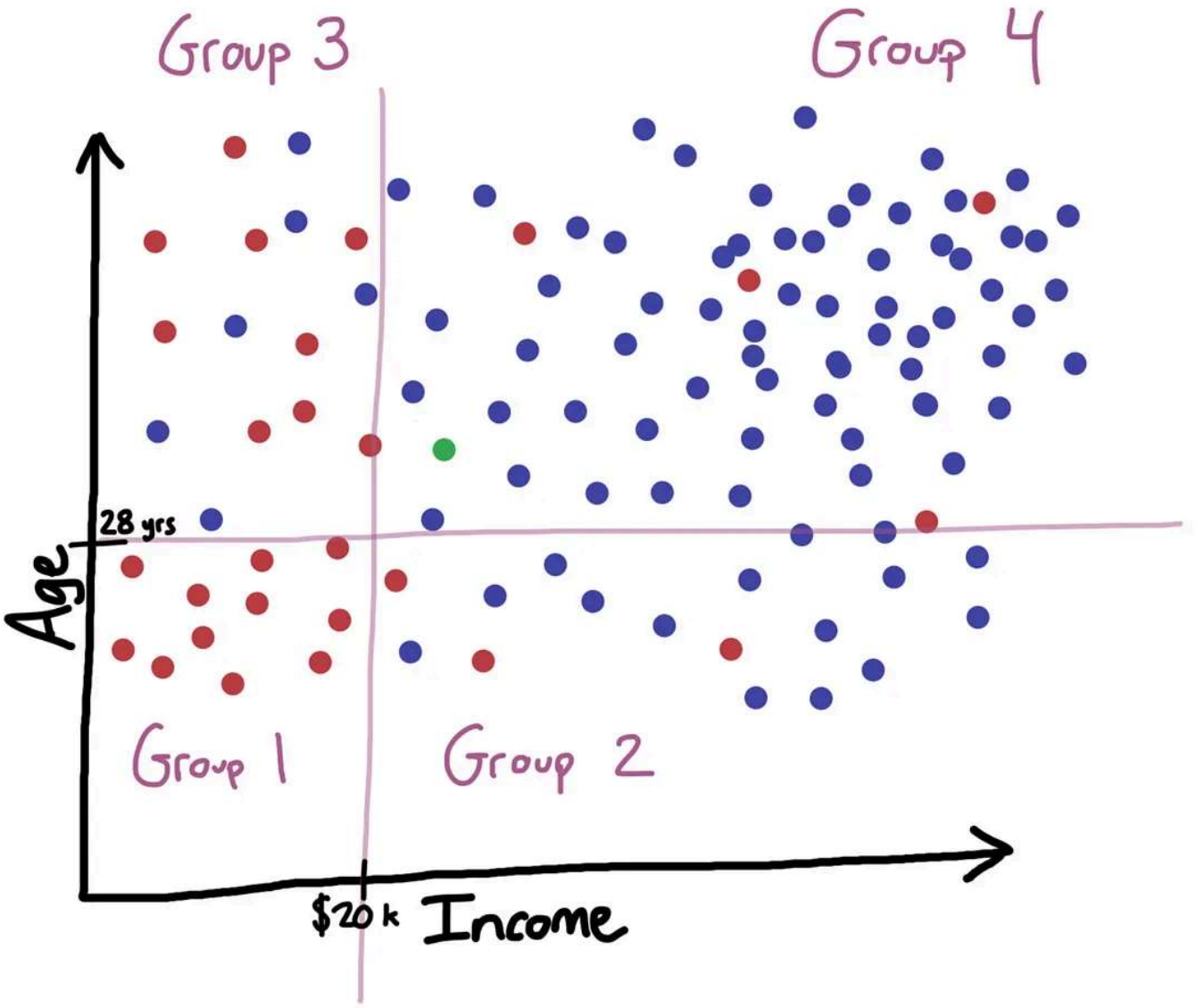

Image created by author

And here's what this second split looks like in tree form:

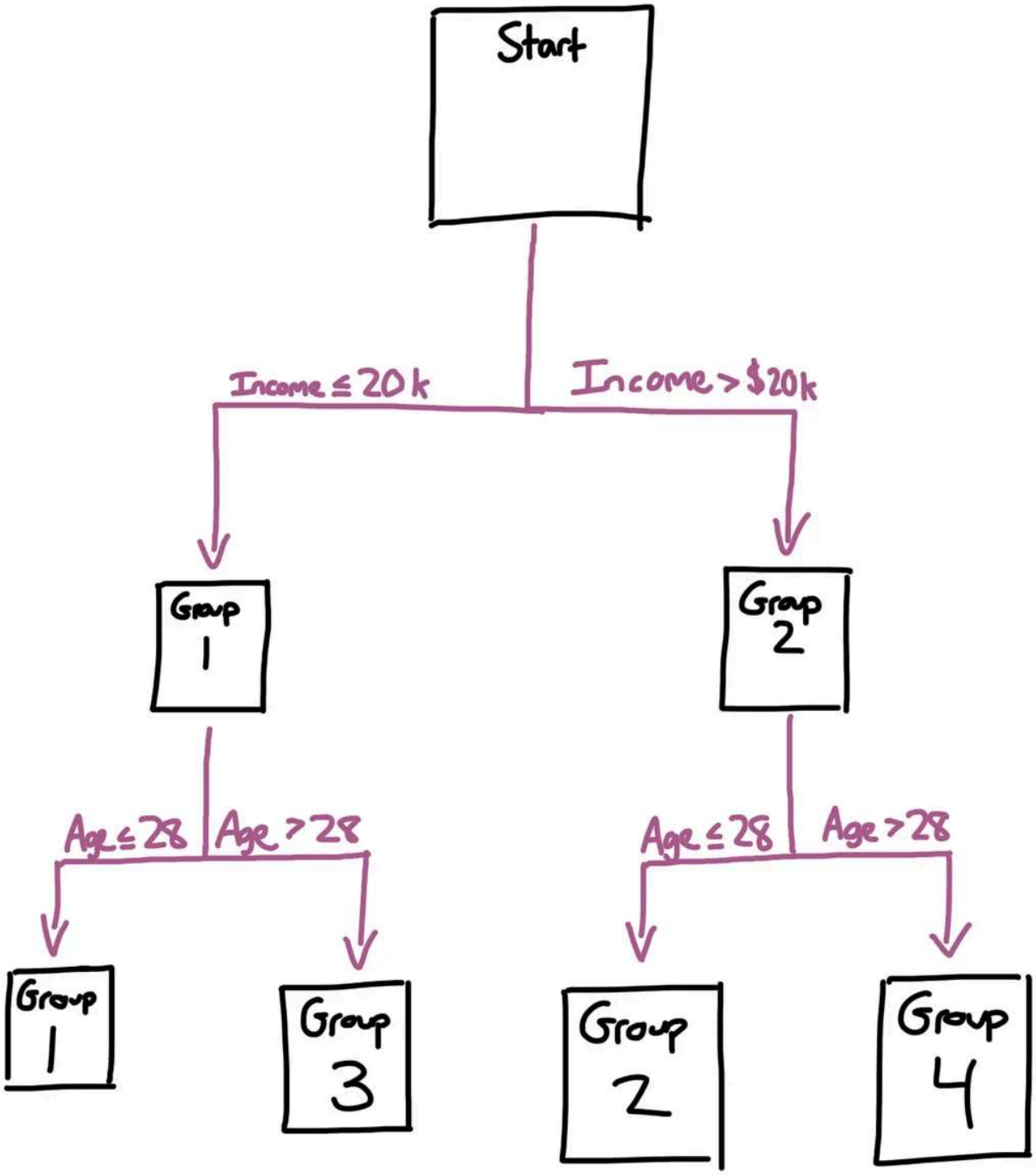

Image created by author

We can keep making smaller and smaller splits like these until we're left with just one outcome–default or no default–on the bottom of each branch. Another way to look at this is that we keep making the boxes on the graph smaller and smaller until each box only contains one color of dots.

For each split, the computer finds the optimal value to split on by using an interesting mathematical process that we won't delve into here.

To test a new case using the decision tree algorithm, you simply start at the beginning and follow the branches down.

Let's say Ted makes $50,000 a year and is 30 years old. We start at the top of the tree.

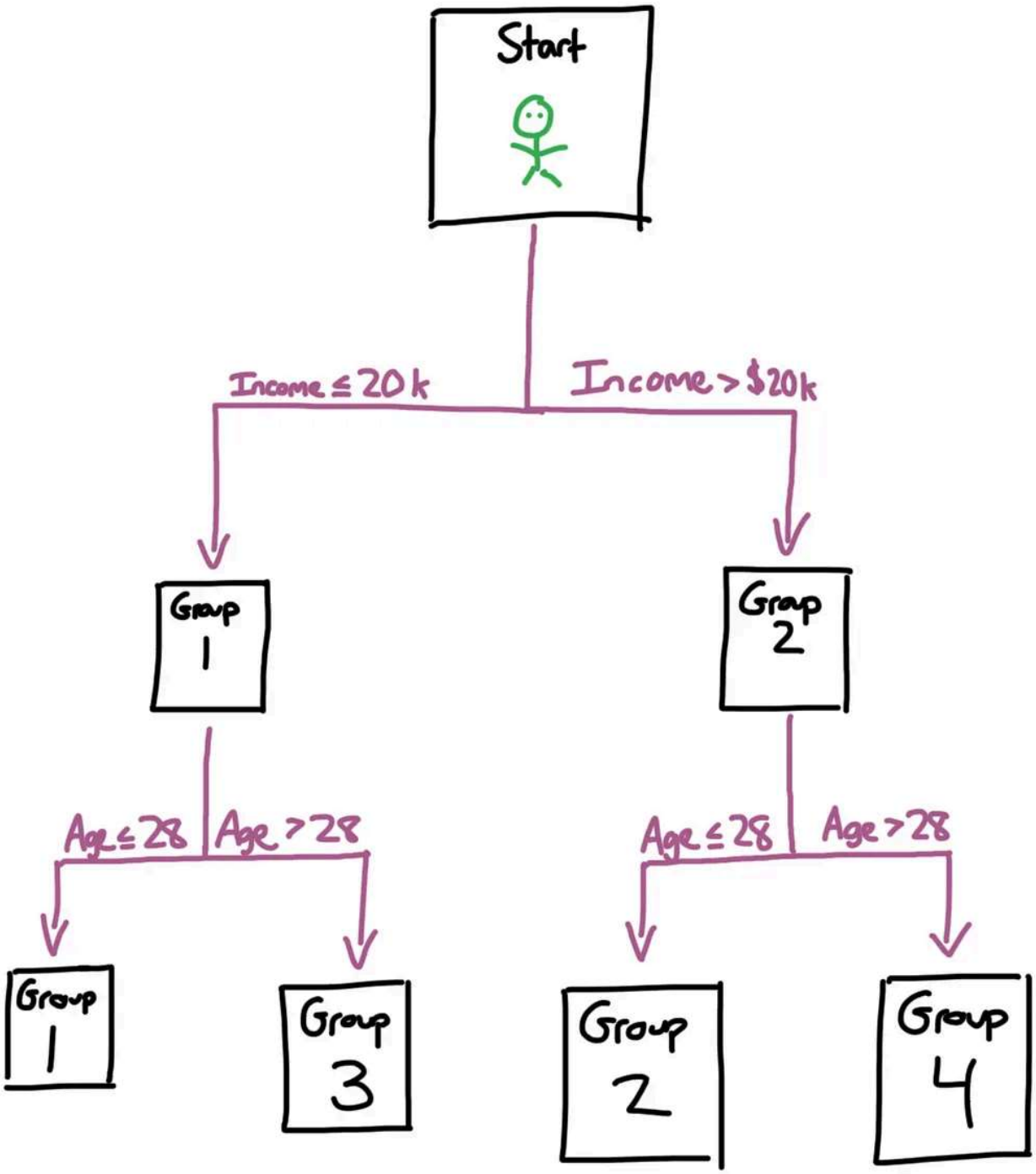

Image created by author

Then, we move down and right, because Ted makes more than $20,000 per year.

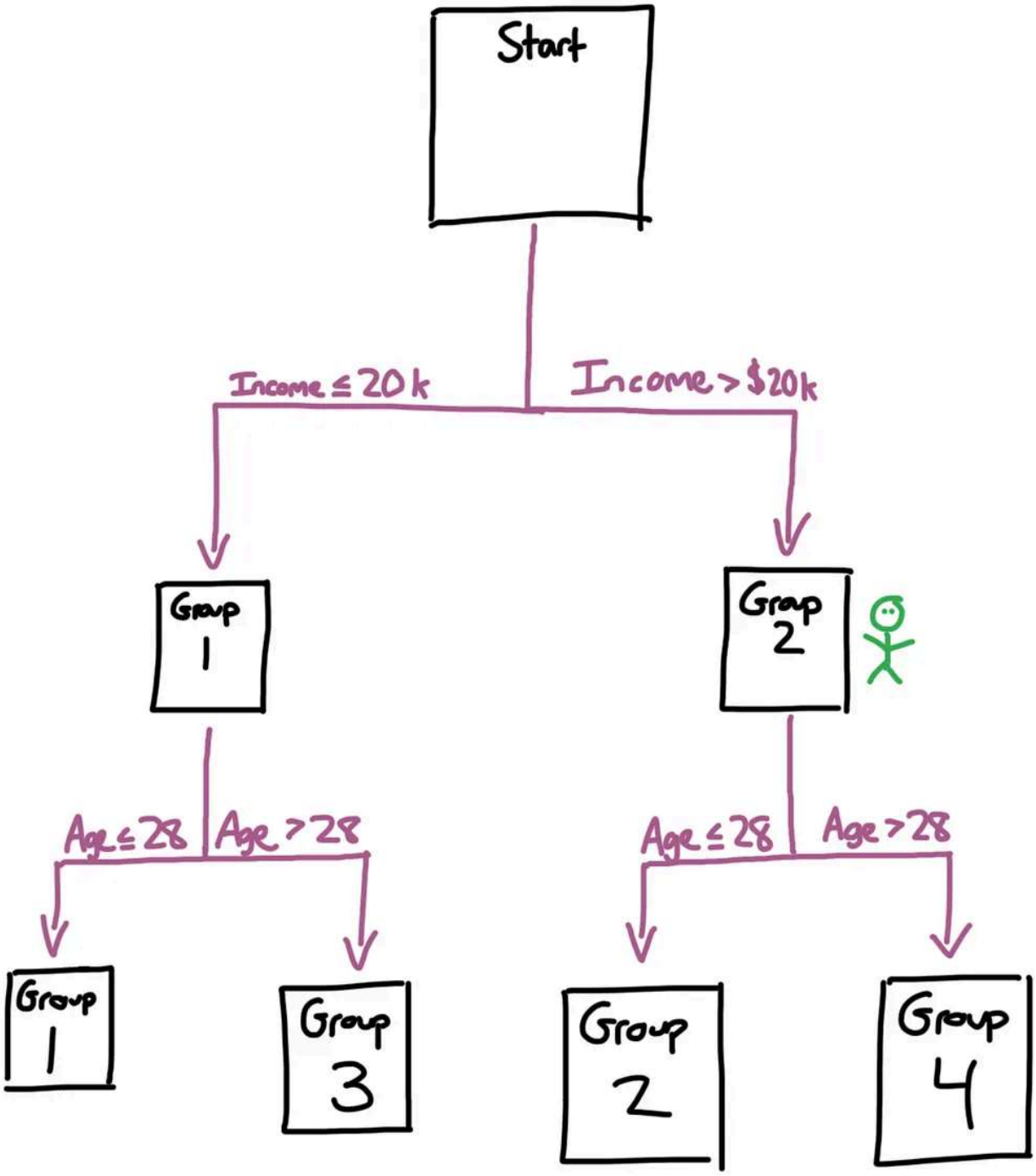

Image created by author

From here, we move to the right because Ted is older than 28.

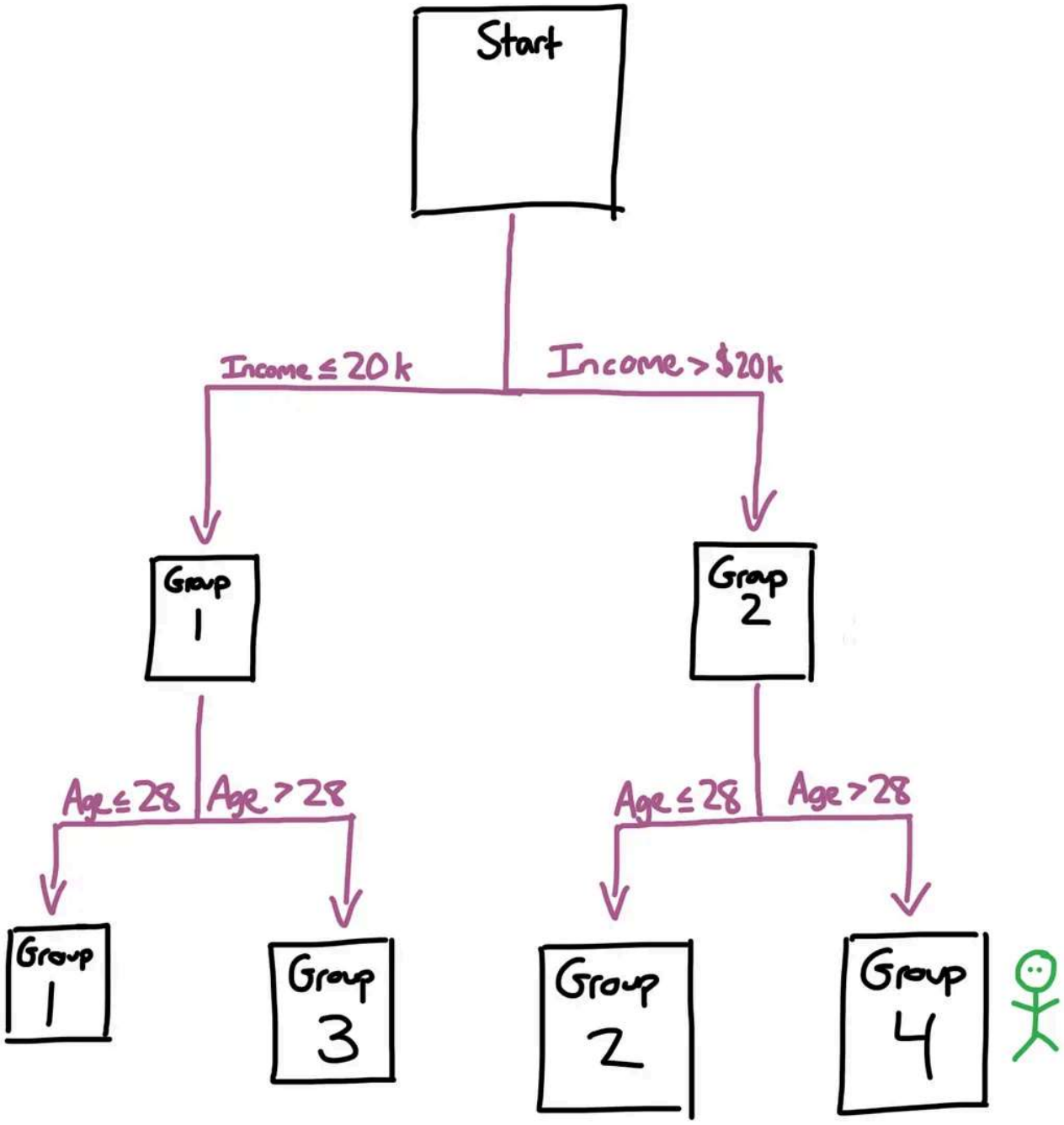

Image created by author

This process continues, until we get all the way to the bottom of the tree, where there is only one possible outcome on a leaf. In a real decision tree, there would be many more branches than the two layers shown here.

The issue with this algorithm is that it returns a binary result: either Ted will default, or Ted won't default. Given that any model is inherently imperfect,

this binary result isn't that useful for credit modeling. We can't use a binary result to set Ted's interest rate.

The solution is to use **random forests**. A random forest is simply a collection of decision trees, all structured slightly differently. To make a prediction, each tree in the forest "votes" for whether or not it thinks Ted will default on the loan. Ted's credit risk is the percentage of the trees that say he will default.

Like the other algorithms we discussed, the decision tree algorithm can be extended to as many dimensions as we want. One advantage of the decision tree algorithm is that it's very good at capturing non-linear data. However, decision trees sometimes look too far into noisy data, thinking that they see patterns where statistically significant patterns don't exist.

There's one more type of machine learning algorithm that we'll briefly touch on here.

Neural Networks

Neural networks are considerably more complex than the three algorithms we've already discussed. They're a fascinating algorithm that's modeled off of the human brain, and they're at the core of most systems that can be considered "AI".

Neural networks are usually overkill for simple credit risk modeling problems, but in certain cases, they can be tremendously useful.

If you want to learn more about neural networks, you can read a detailed explanation of them here.

Summary of the risk modeling process

These four classes of algorithms (k-nearest neighbors, logistic regression, decision tress, and neural networks) are just the beginning of the machine learning used in credit risk modeling. But understanding the basics of these algorithms gives you a bit of insight into the whole credit risk modeling process, and this understanding gives you the perfect springboard to learn more about machine learning.

First, we developed a bit of domain knowledge about how loans and interest rates are used in the real world. Then, we looked at splitting the data into train and test sets to analyze various models. Finally, we analyzed several models that we could test out, and we know that we'd use the most effective models' outputs to make our predictions.

A similar process is applicable to almost any problem that involves machine learning.

First, it's important to develop some domain knowledge about the problem you're dealing with so that you know how to ask the right questions. Then, you have to clean and prepare your data (a topic we didn't delve into here). Next, you train and test a variety of models, honing the models to maximize prediction accuracy. Finally, you present your results to others and apply them to real world problems, repeating this process continuously.

If you have a better understanding of this general process, then you're well on your way to better understanding credit risk modeling and machine learning as a whole!