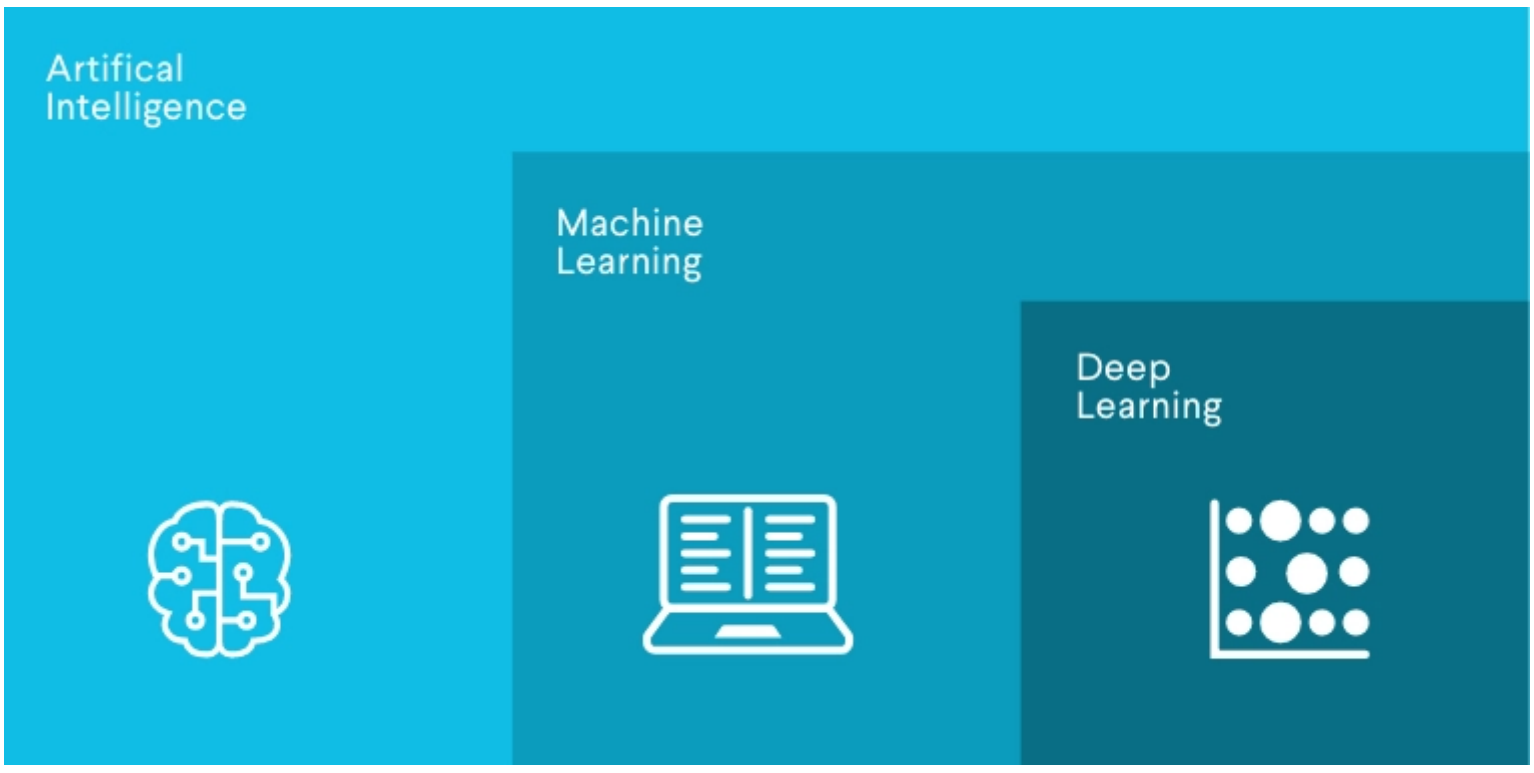
RAVELIN INSIGHTS

Machine learning for fraud detection

Everything you need to know about models, neural networks, risk scores, thresholds and adding human insight

CONTENTS

What’s the difference between artificial intelligence and machine learning?



Artificial intelligence (AI) has been in sci-fi movies for nearly 100 years, in films like RoboCop, The Matrix, Star Wars and The Avengers. In reality, today’s AI is not quite the same as its portrayed in the movies… yet. But there is some truth in that AI is like computers acting with human intelligence.

Machine learning is a subset of AI, and the key difference is the ‘learning’. With machine learning, we are able to give a computer a large amount of information and it can learn how to make decisions about the data, similar to a way that a human does.

Machine learning has many uses in our everyday lives - for example email spam detection, image recognition and product recommendations eg. for Netflix subscribers.

Deep learning is a subset of machine learning. The key advantage deep learning gives is the ability to create flexible models for specific tasks (like fraud detection). With traditional machine learning, we couldn’t create bespoke models as easily - we’ll explain why this is so important later on.

Machine learning is a set of methods and techniques that let computers recognise patterns and trends and generate

Old school fraud detection

Traditionally businesses relied on rules alone to block fraudulent payments. Today, rules are still an important part of the anti-fraud toolkit but in the past, using them on their own also caused some issues.

False positives

Using lots of rules tends to result in a high number of false positives - meaning you're likely to block a lot of genuine customers. For example, high-value orders and orders from high-risk locations are more likely to be fraudulent. But if you enable a rule which blocks all transactions over $500 or every payment from a risky region, you'll lose out on lots of genuine customers' business too.

Fixed outcomes

The thresholds for fraudy behaviour can change over time - if your prices change, the average order value can go up, meaning that orders over $500 become the norm, and so rules can become invalid. Rules are also based on absolute yes/no answers, so don't allow you to adjust the outcome or judge where a payment sits on the risk scale.

Inefficient and hard to scale

Using a rules-only approach means that your library must keep expanding as fraud evolves. This makes the system slower and puts a heavy maintenance burden on your fraud analyst team, demanding increasing numbers of manual reviews. Fraudsters are always working on smarter, faster and more stealthy ways to commit fraud online. Today, criminals use sophisticated methods to steal enhanced customer data and impersonate genuine customers, making it even more difficult for rules based on typical fraud accounts to detect this kind of behaviour.

Rules and machine learning are complementary tools for fraud detection

Although machine learning has delivered a huge upgrade to fraud detection systems, it doesn't mean you should give up using rules completely. Your anti-fraud strategy should still include some rules where it makes sense, and also incorporate the benefits of machine learning technology.

Why is machine learning suited to fraud detection?

Super fast

When it comes to fraud decisions, you need results FAST! Research shows that the longer a buyer's journey takes the less likely they are to complete checkout.

Machine learning is like having several teams of analysts running hundreds of thousands of queries and comparing the outcomes to find the best result - this is all done in real-time and only takes milliseconds.

As well as making real-time decisions, machine learning is assessing individual customer behaviour as it happens. It's constantly analyzing 'normal' customer activity, so when it spots an anomaly it can automatically block or flag a payment for analyst review.

Scalable

Every online business wants to increase its transaction volume. With a rules only system, increasing amounts of payment and customer data puts more pressure on the rules library to expand. But with machine learning it's the opposite - the more data the better.

Machine learning systems improve with larger datasets because this gives the system more examples of good and bad eg. genuine and fraudulent customers. This means the model can pick out the differences and similarities between behaviors more quickly and use this to predict fraud in future transactions.

Efficient (and cheap!)

Remember that machine learning is like having several teams running analysis on hundreds of thousands of payments per second. The human cost of this would be immense - the cost of machine learning is just the cost of the servers running.

Machine learning does all the dirty work of data analysis in a fraction of the time it would take for even 100 fraud analysts. Unlike humans, machines can perform repetitive, tedious tasks 24/7 and only need to escalate decisions to a human when specific insight is needed.

More accurate

In the same way, machine learning can often be more effective than humans at uncovering non-intuitive patterns or subtle trends which might only be obvious to a fraud analyst much later.

Machine learning models are able to learn from patterns of normal behavior. They are very fast to adapt to changes in that normal behaviour and can quickly identify patterns of fraud transactions.

This means that the model can identify suspicious customers even when there hasn't been a chargeback yet. For example, a neural network can look at suspicious signals such as how many pages a customer browses before making an order, determine whether they are copying and pasting information by resizing their windows and flag the customer for review.

How does a machine learning system work?

We use a few different forms of machine learning at Ravelin - here's a simple explanation of how a supervised machine learning system works. Listen to this podcast to hear more detail about the process.

How a machine learning system works:

Input data

Extract features

Train algorithm

Create model

Input data

When it comes to fraud detection, the more data the better.

For supervised machine learning, the data must be labelled as good (genuine customers who have never committed fraud) or bad (customers with a chargeback associated with them or have been manually labelled as fraudsters).

Extract features

Features describe customer behaviour, and fraudulent behaviours are known as fraud signals.

At Ravelin, we group features into five main categories, each of which has hundreds or thousands of individual features:

Identity

Number of digits in the customer’s email address, age of their account, number of devices customer was seen on, fraud rate of customer's IP address.

Orders

Number of orders they made in their first week, number of failed transactions, average order value, risky basket contents.

Fraud rate of issuing bank, similarity between customer name and billing name, cards from different countries.

Locations

Shipping address matches the billing address, shipping country matches country of customer's IP address, fraud rate at customer’s location.

Network

Number of emails, phone numbers or payment methods shared within a network, age of the customer’s network.

Train algorithm

An algorithm is a set of rules to be followed when solving complex problems, like a mathematical equation or even a recipe. The algorithm uses customer data described by our features to learn how to make predictions eg. fraud/not fraud.

In the beginning, we’ll train the algorithm on an online seller’s own historical data, we call this a training set. The more fraud in this training set the better, so that the machine has lots of examples to learn from.

Create a model

When training is complete you have a model specific to your business, which can detect fraud in milliseconds.

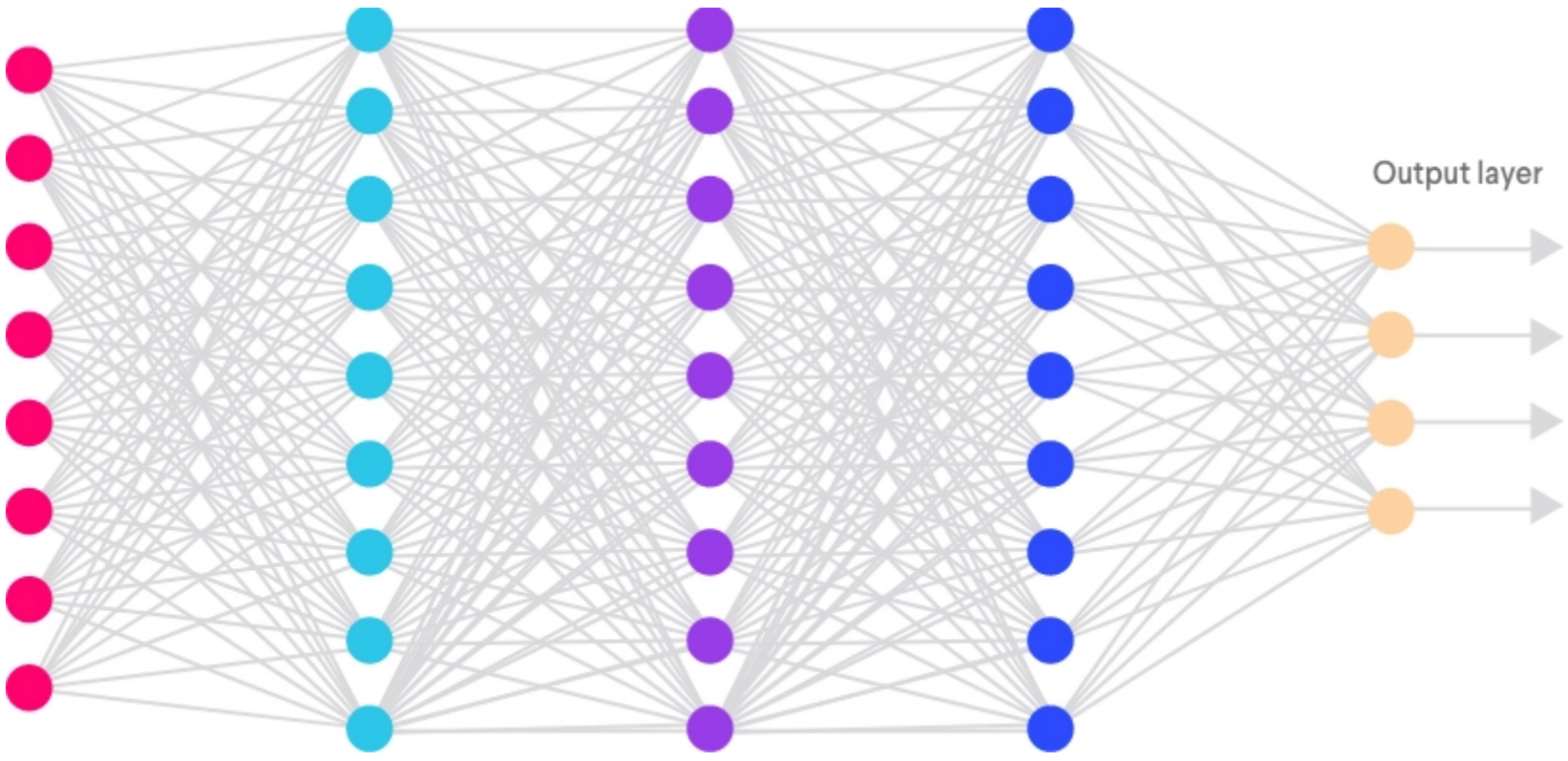
We constantly keep an eye on the model to make sure it is behaving as it should, and we’re always looking for ways to improve it. We regularly improve, update and upload a new model for every client so that the system will always detect the latest fraud techniques.

Deep neural networks and a micro-model architecture

At Ravelin, We use deep neural networks - a neural network is a machine learning model architecture loosely inspired by the

greater accuracy.

Neural networks have deep layers of machine models



We use a mixed-data approach for our neural network architecture. This means we create separate neural networks that focus on different aspects about the customer: their behaviour (eg. the average number of orders per week), the natural language associated with them (eg. the order items, email address), the anomalousness of their activity (eg. whether they proceed through checkout 20x faster than the average customer) or any image data associated with them (eg. profile picture).

We combine the layers in these networks into a singular model. This final neural network is trained to learn which aspects about the individual business's customers are most important for detecting fraud.

How can you tell the model is working?

After the training, to check that the model is working correctly, we show the model some data which it has never seen before, but which we know the fraud outcomes for. If the model detects the fraud correctly, we can deploy it to be used against the online business's transactions. We also do some automatic common-sense analysis on recent data for which we do not have fraud labels to ensure the model will behave correctly when it is deployed.

There are certain fraudy situations which the model should always pick up on - some examples are:

- High velocity of new payment methods eg. a customer adds new 10 payment cards in an hour
- Suspicious email address eg. a mismatch between the account name or name on the card, or rude/naughty words in the

- Orders from a particularly fraudy location, shipping to a known fraud hotspot or a PO Box rather than a residential address

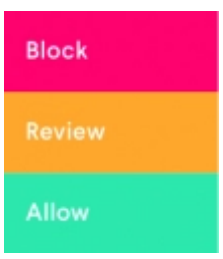
All of these examples should be flagged as fraudy - so what happens when the machine makes a prediction?

Using machine learning to generate a fraud risk score

Customer places order

ML generates features

Model predicts risk score



At the point of the transaction, the model gives each customer a risk score on the scale of 1-100. The higher the score, the higher the probability of fraud.

You can choose what level of risk is right for your business, and set thresholds for what proportion of transactions you want to allow, block and manually review or challenge using 3D Secure.

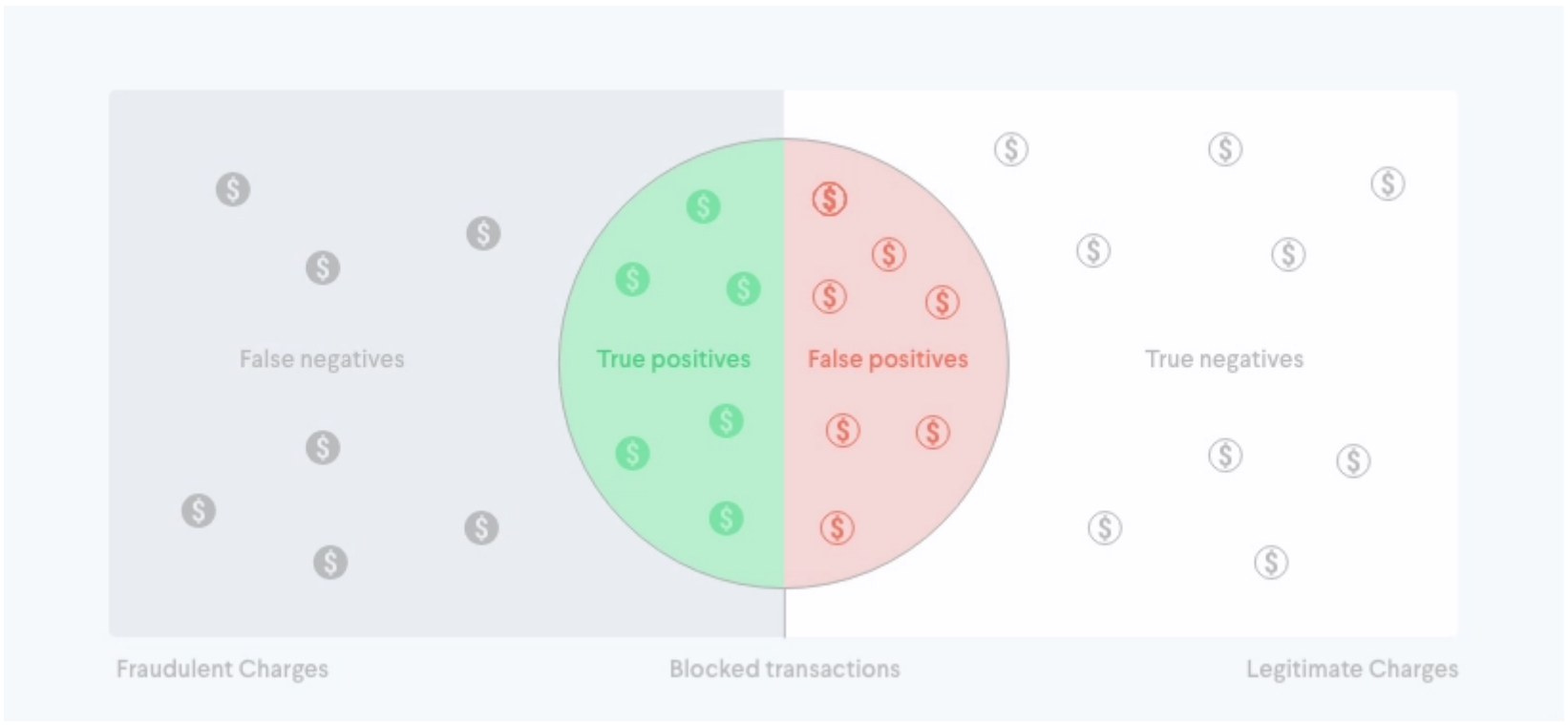
Setting the right risk threshold for your business

The next step is to ask yourself, where on the scale is the right risk threshold for my business?

Threshold analysis - precision and recall

Determining the right risk threshold involves doing data analysis based on the principles of precision and recall. It's a complicated balancing act between:

- True positives (how many fraudsters we block)
- False positives (how many good people we block)
- False negatives (how many fraudsters we allow)



Scale is very important here. For context, the typical acceptance rate for a Ravelin client is usually higher than 98 or 99%, so almost all transactions are approved. It's within the small band of rejected transactions where the optimization occurs.

> Risk analysis asks how close to 100% acceptance you can get without the cost of fraud becoming too high

The right level of risk is individual to each business. A business with a high volume of low-value transactions (eg. food delivery) may set the risk threshold very high, so that they can ensure they are blocking the least possible amount of genuine transactions.

Our investigation analysts are experts in these calculations and can help you find the right thresholds to suit your risk appetite.

Does your business need its own machine learning model?

It's always best to use your own customer data for your business as it will be the most accurate at detecting fraud within your future customers. Different business models can have very different customer order cycles and amounts - for example it could be normal for someone to order from a food delivery business every day, whereas this would be very unusual for online clothes sales.

There are also huge variations in other aspects, for example it might take customers only a few minutes to order from a ticketing site, but a taxi app order can take as long as the ride lasts.

Unlike other fraud providers, we build a 100% personalised model for each of our clients, so predictions will be based on fraud signals in their customer base alone. This stops the model being swayed by patterns in unrelated industries, creating more specific predictions and better performance.

There’s always the chance that an online business might not have enough data to train their own model right away. A business might have a very low sales volume, mainly sell through affiliates, or sometimes the logging simply hasn’t been set up to collect the data in the right format.

It’s no problem if you don’t have enough data to train your own model right away. To get your business up and running quickly, we’ll use a generic model based on historical fraud patterns we’ve seen before. We don’t share any of the customer data between businesses, but we can re-use the algorithms we’ve already trained. This makes it easy for us to pick the right components off the shelf and it means you can start using a model to detect fraud in just one week.

Because we use this micro-model approach, we can pick and choose the ones which are most suitable for your individual business to make up a semi-customised larger model. As soon as the model starts working on your data it will begin to adapt and tailor to your customer base, and therefore become more effective. The model improves as we give it more data, chargebacks and manual reviews.

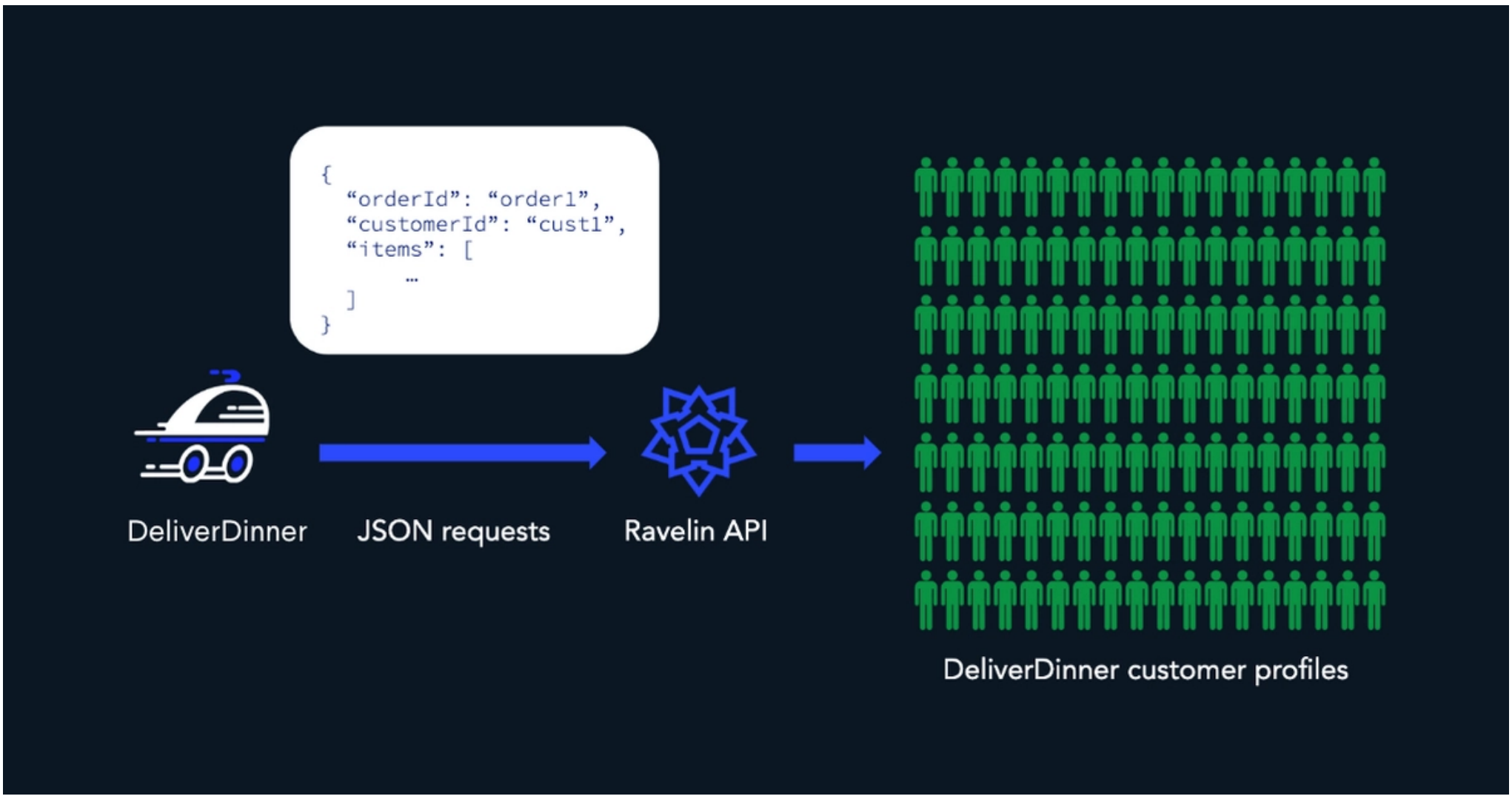
Why it’s important to use historical data and not just recent data

We’ve found that within a month we have chargebacks for around 30% of fraud - that means up to 70% of fraud hasn't been recorded yet. This means that if we used only the most recent data, the model wouldn't be able to distinguish the hidden fraudsters (who haven’t made a chargeback yet) from the rest of recent genuine customers. **Listen to Milly, a Ravelin Data Scientist, talk about this in more detail in this webinar.**

An example of how we build a model

Let’s imagine we’re building a machine learning model to detect fraud for a food delivery business. Our fictional business is called **DeliverDinner.**

When **DeliverDinner** joins Ravelin as a new client, they start to send live transaction traffic to our API.

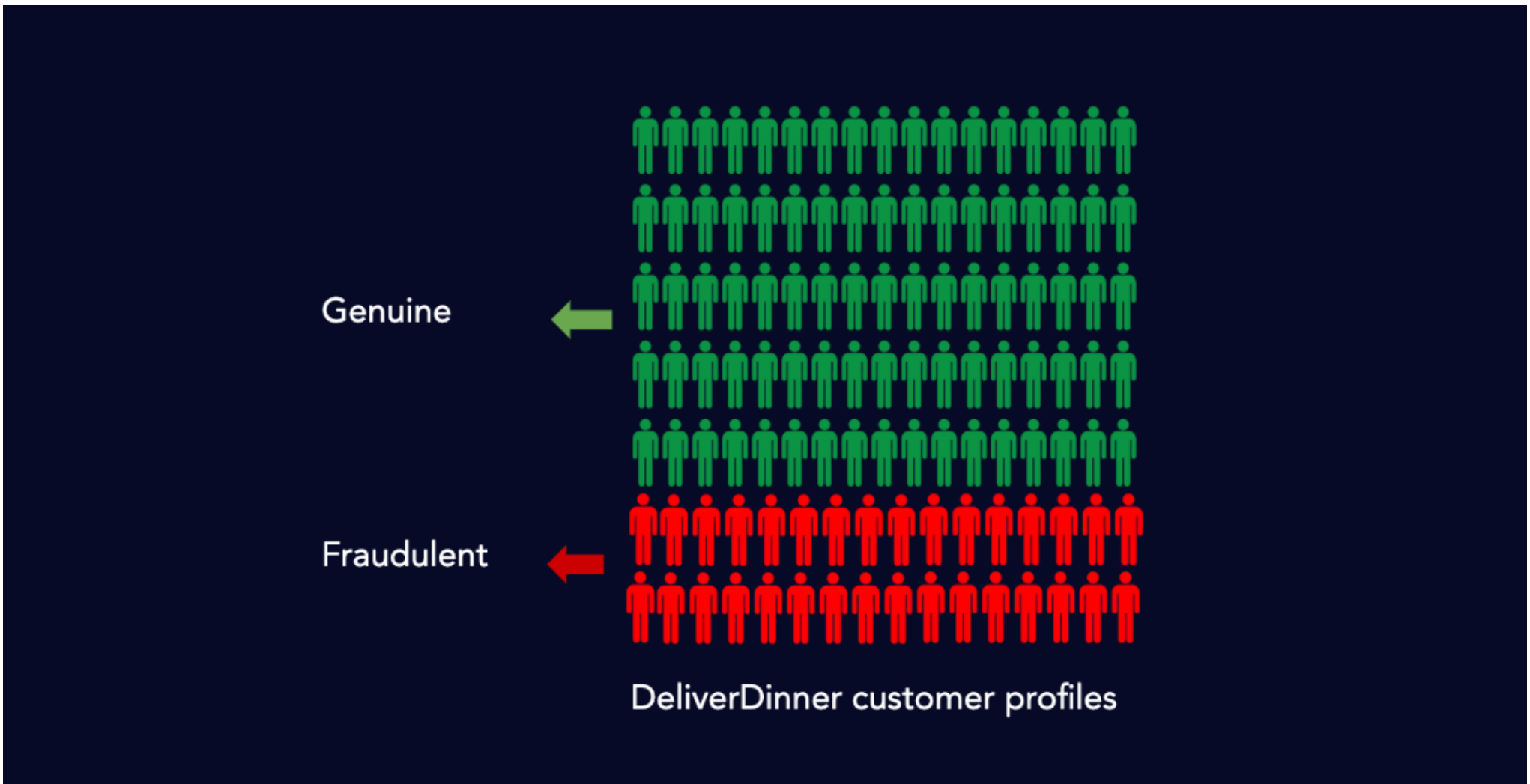


Every time a customer registers, adds an item to their basket, or does anything on the DeliverDinner website, it sends a JSON request to the API. This means we store lots of data about DeliverDinner customers and everything they've ever done in their account. We bundle these into customer profiles.

To use this data for machine learning we need to do three things:

1. Label the customers as fraud/not fraud
2. Describe the customers in computer language
3. Train the model

Step 1: assign labels



We look at any customer which has had a chargeback or which has been manually reviewed as fraudulent by the merchant - and label them as fraud.

Step 2: create features

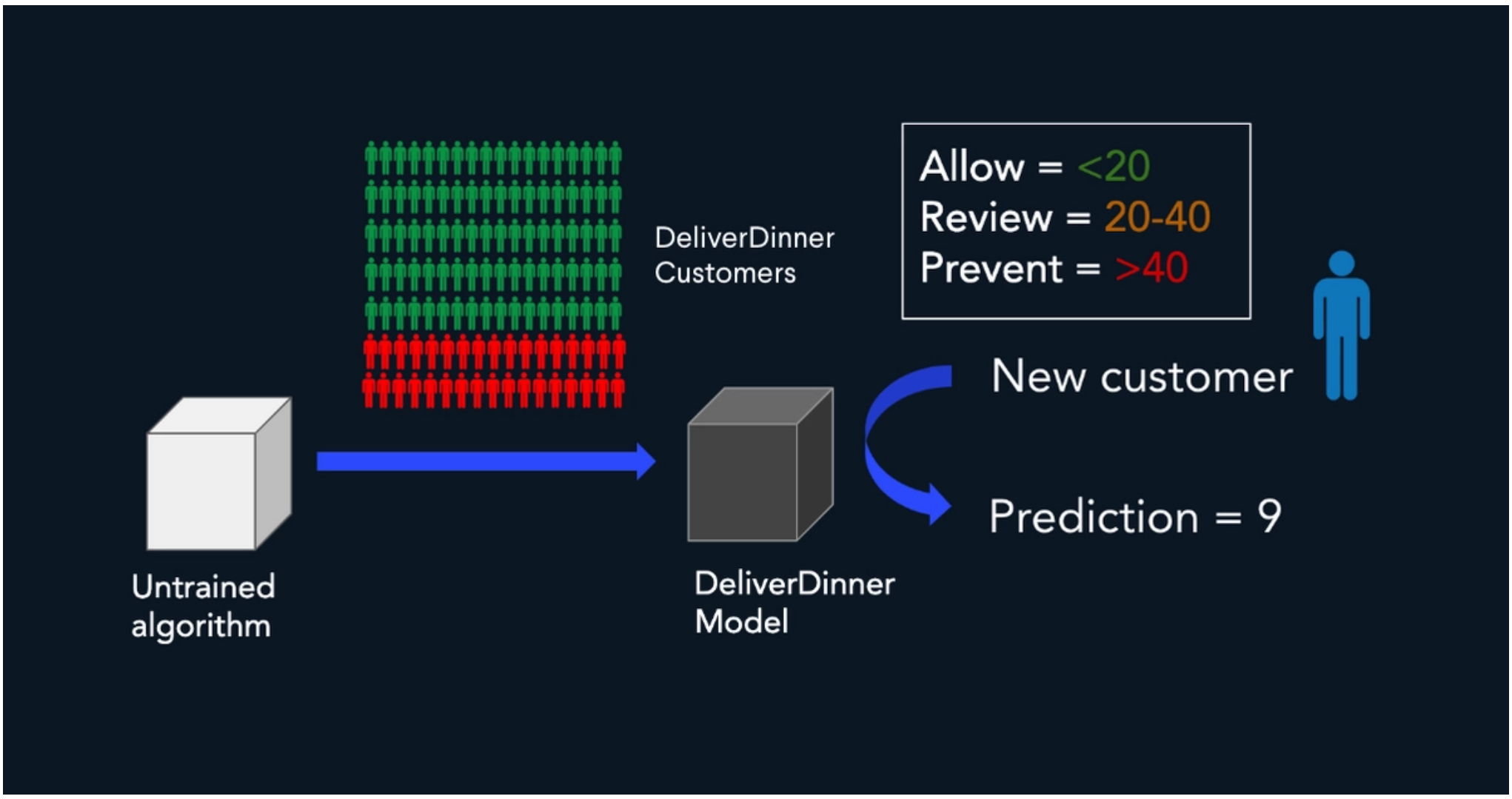
Creating features is basically describing each customer in a way that the computer can understand. We want to describe the characteristics of a customer which indicate if they would be fraudy or genuine - this is based on the same aspects that a fraud analyst would look at to make the decision.

Examples of features which could be good indicators of fraud are:

- **Order rate** - fraudsters order at a much more rapid pace, we quantify this as number of orders per week.
- **Email** - fraudster might have a dodgy-looking email, we quantify as % of digits in the email address
- **Delivery location** - it could be somewhere typically genuine/unlikely to be fraud like a penthouse apartment, or it could be somewhere fraudy like a park. We quantify this as the location fraud rate %

All features are created as a number as the model can’t absorb raw text. We build up our features and categorize them into groups.

Step 3: train the model



We need to feed the algorithm the data so that it can learn how to solve the problem. At this stage, we feed in the training data.

The training data is a bunch of DeliverDinner data about customers, described in terms of their features and labels to let the algorithm know if they are a fraudster or a genuine customer. This helps the model learn how to tell the difference between genuine/fraudulent.

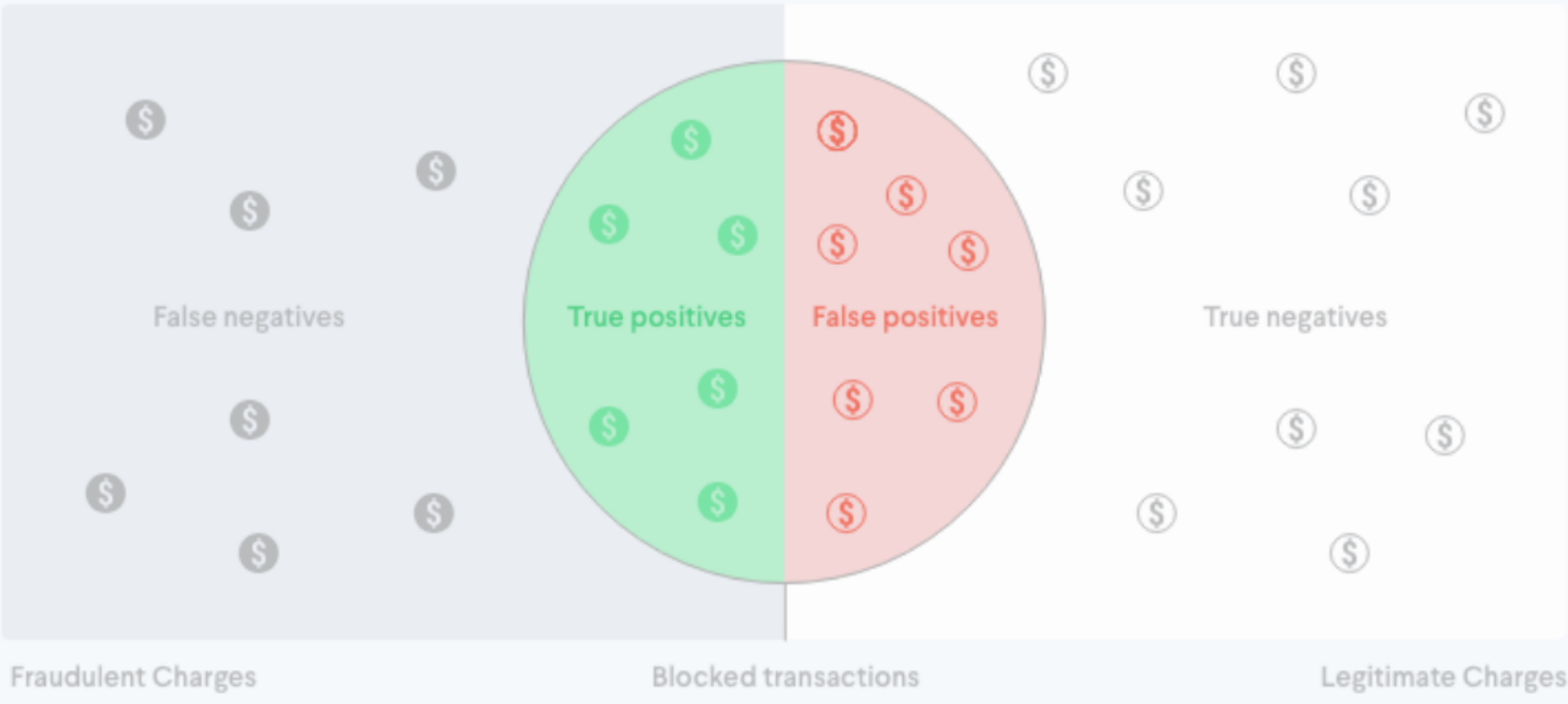
Within DeliverDinner's dataset, this might show that genuine customers tend to order around once a week, they tend to use the same card each time and the billing + delivery address are often the same. Fraudsters might show that they order several times a week, use lots of different cards, that their cards have failed registration and that the billing and delivery address don't often match.

The algorithm will take this at face value, and learn the perfect way to check if a customer features look more like the genuine customer pile or the fraudulent customer pile.

When we show the model a new customer it hasn't seen before, it compares it with the genuine/fraudy customers it has seen before and produces a fraud score. This score represents how likely the new customer is to be fraudulent.

For the majority of customers, the fraud score will be quite low, as there are many more genuine customers than fraudsters. When it's a low score, we recommend allowing the customer and the transaction to go through. If it's a medium score, we recommend a Review of the transaction, eg. sending the customer a 3D secure challenge to authenticate. If the score is very high we'd recommend blocking the customer from making the transaction.

How do you decide the limits to allow/review/prevent?



Precision asks: of all the prevented customers, what proportion were fraudsters?

Recall asks: of all the fraudsters, what proportion did we prevent?

Putting precision & recall in context

If your prevent threshold is at 95, you're blocking a very small % of customers. You'd have very high precision - you're only blocking a few customers that you're fairly sure are fraudsters. You'll have a very low false-positive rate. However, recall is likely to be low as there are likely to be fraudsters with scores under 95 which you're not preventing.

If we look at the opposite situation - if you have a block threshold of 5. You're preventing a huge amount of your traffic and so you're likely to have very poor precision - and probably end up with lots of false positives. You will have high recall - as you're going to block most if not all of the fraudsters.

Setting the right risk threshold

It's a bit of a balancing act between the two, and where you set your thresholds depends on your individual business priorities. It's easy to tweak these depending on your risk appetite, or if you are more concerned about chargebacks or false positives.

Understanding precision, recall and setting risk thresholds is important for us to understand how we can assess our model accuracy and make sure it is improving.

How we select the right business data for feature engineering

So we know how we build a model, but how do we decide what data it should look at?

Every business has a lot of data, but not all of it is relevant for fraud. Here's how we select specific data 'features' to analyze and get an indication of fraud.

First, what is a feature and how is it engineered?

At a basic level, a feature is an individual measurable property or characteristic, such as the cost of a transaction. Feature engineering is the process of extracting these meaningful characteristics to use as learning material for the algorithm.

Building features

We look for features to capture certain aspects that help us predict fraud. We group the types of features into the below categories.

Traditional features

These are the typical aspects that predict fraud, for example orders, transactions, cards, location, email. These features generally cover the data you would expect to find on your receipt and are customer-centric.

Behavioral features

We derive behavioral features from the customer session - these are features are based on describing the customer actions eg. velocity of orders, time spent on the page, length of time between adding a new card and making an order. One purpose of extracting these features is to capture other subversive technology use eg. if a fraudster is using a script to scrape a webpage vs normal browsing activity.

Real-time features

Real-time features are based on the up to date, real-world incidences of fraud. These features are all based on categorical data - give the real-time rate of fraud by category eg. country / ASN card digits / email domain etc. An example feature could be the fraud rate in certain regions/countries.

One purpose of these features is to help merchants to expand into new markets where they have no existing data. We monitor the real-time traffic to help our merchants seamlessly move into new markets, without seeing any adverse effects from the machine learning models eg. bias.

Individual customer features

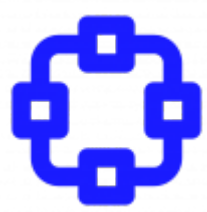
These features tell us about the similarity with the specific customer's typical past behavior. This could be their typical spend, their regular billing address, home IP address etc.

Session tracking features

These features are a little more involved than the behavioral features. These features cover the data we get from Javascript eg. whether the customer is pasting a card number into the checkout, cookies, if they are using a password vault etc. One purpose of these features is to capture genuine customer behavior eg. taking time to change the size of a piece of clothing.

Entity features

We divide features into customer-centric and entity-centric. Entities are things like devices, addresses, locations, domains and emails. An example feature is the number of orders shipped to a certain address. One purpose of these features is to alert us to a fraud goods drop-off point



Network derived features

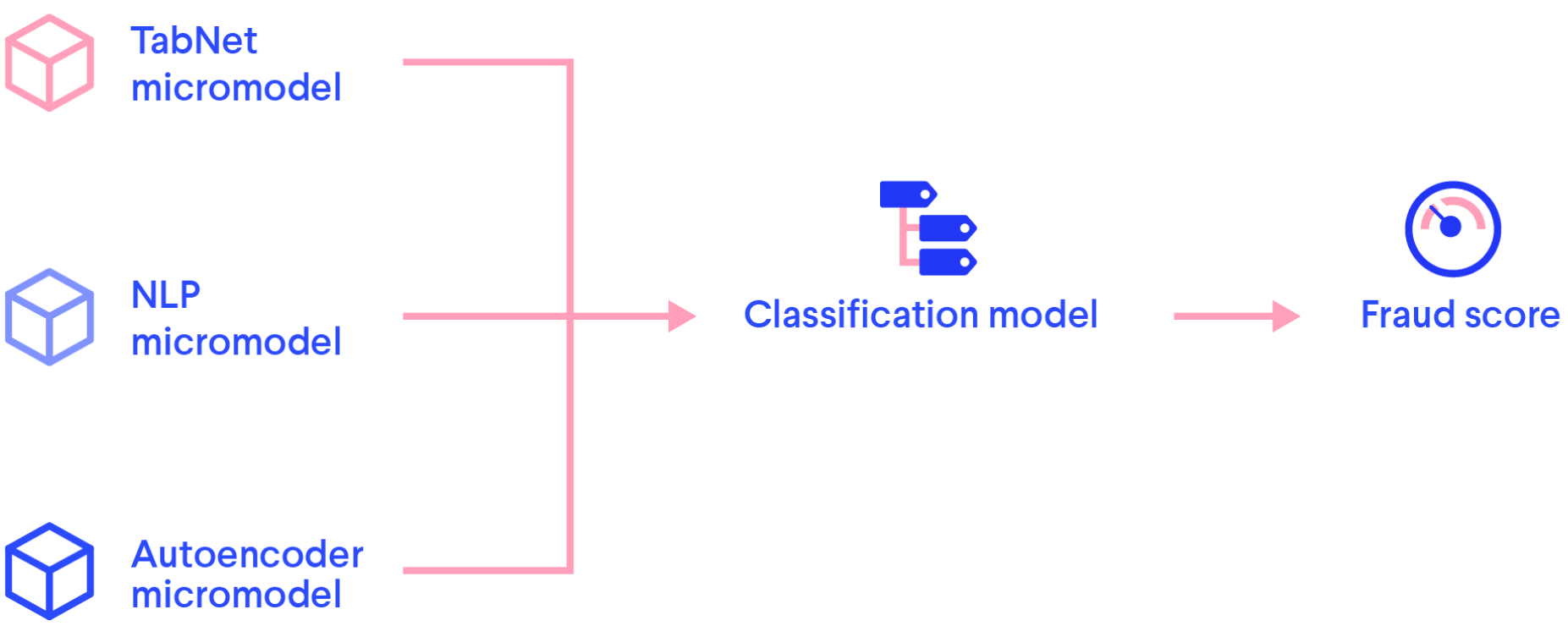
As well as customer-centric and entity-centric features, we also look for network level features. These features focus on network topology (network shape) as a means of enhancing our customer data. An example is account sharing between a family in the same house vs. account takeover where networks of hundreds of accounts use the same few devices.

We use a combination of different model architectures to get the best performance - here's a short introduction to three of the prominent architectures we use and why we use them...

Because each business is unique, we train and assemble individual micromodels for every one. This results in each client having a solution suited to their specific data and fraud trends.

The solution is made up of a micromodel architecture - meaning it's a combination of different models for different data in one. Each individual business will have a different weighting on the different models depending on their data.

These individually specialized models learn the best representations of specific parts of the dataset. Here's a simple representation of this:



Three expert micromodels we use

Three prominent micromodel architectures we use are:

- Tabnet
- Natural Language Processing (NLP)
- Anomaly detection autoencoders

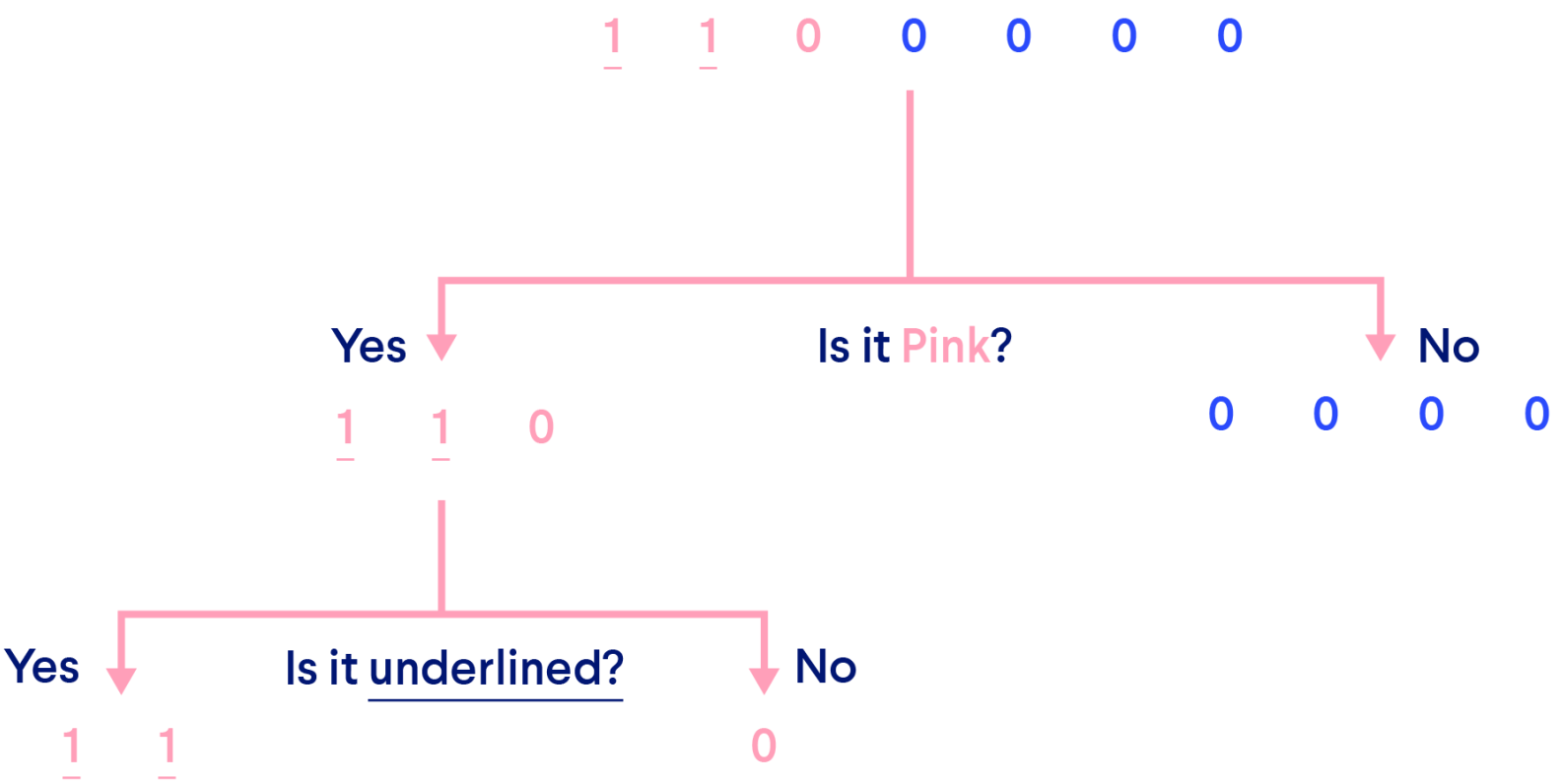
Besides these three models, we use a variety of customisable experts to build a ‘mixture of experts’ model. Each of the models is a specialist in its own right - we’ll explain the three above in more detail…

TabNet micromodel

Random forest & decision trees

A random forest architecture is made up of many decision trees - so first, what is a decision tree?

A decision tree is a series of questions with yes/no answers which aims to divide data into different classes. The goal is for the tree to divide the data into groups which are as different from each other as possible, and for the members of each group to be as similar to each other as possible.



Random Forest is an ensemble of decision trees which are blended at the classification level.

Advantages for fraud detection

Random forest architecture is already commonly used in fraud detection. The benefits of this architecture are:

- Rapid deployment
- Multiple trees reduce overall bias
- Easy to interpret and understand the internal workings
- Stable algorithm - not disrupted by adding a new datapoint

Disadvantages for fraud detection

Random forest is blended at the classification level, not representation level. This means the classification model doesn't know which trees carry more/less weight in the decision, which gives it less predictive power.

One problem with random forest is that it can be prone to overfitting. Overfitting occurs when the model or the algorithm fits the

This means it is learning too much about specific training customers vs. performing a more general classification which can apply to many customers. This can lead to poor performance in the live environment on customers that aren't in the training dataset.

Why is TabNet an improvement?

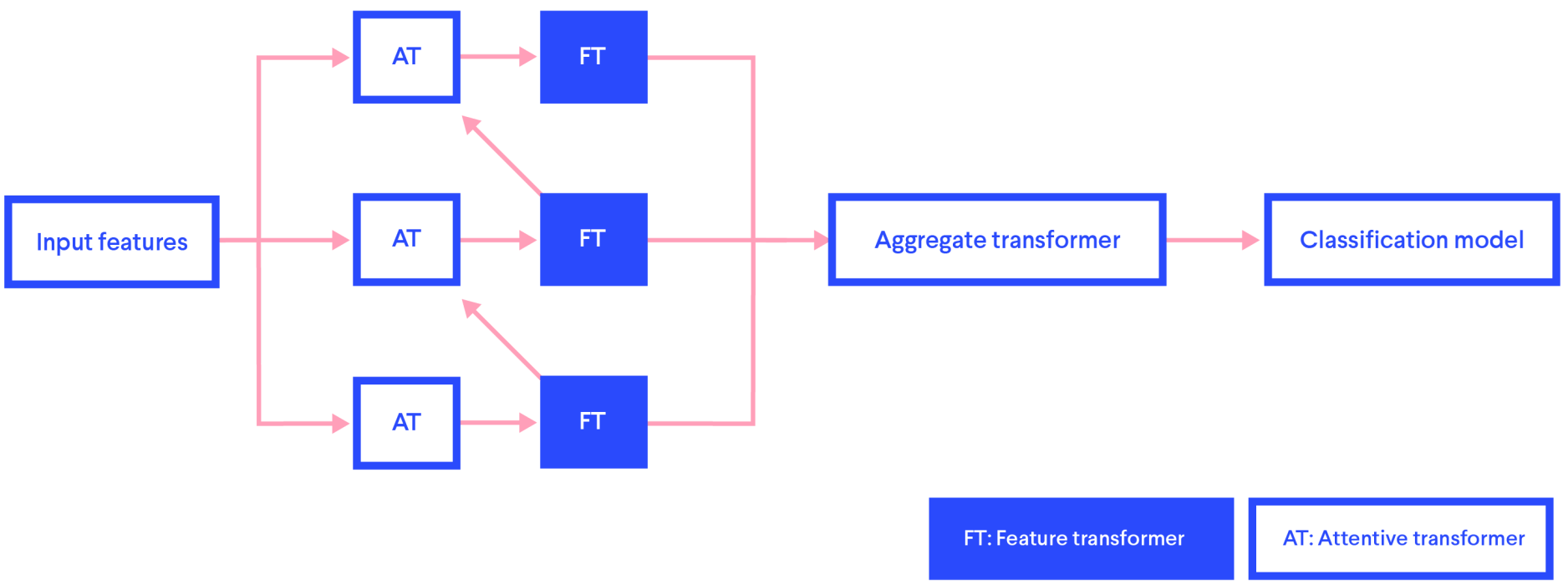
TabNet is great for explainability - it allows us to understand which features carry the most weight on the prediction and enables further analysis - unlike the random forest.

The purpose of this micromodel for Ravelin is to learn about:

- Numerical data
- Low cardinality categorical features (features with low number of categories eg. continents rather than countries)

TabNet results in a high performance model which outperforms other neural network and decision tree variants. What makes it so efficient?

Tabnet model design



More than yes/no

Another key difference with TabNet is that it is not just a yes/no decision and answer. Instead of simply classifying data into true/false classes at each stage, the model can classify according to a value eg. transaction amount. It has two key elements below.

The attentive transformer directs the model's attention. It's a powerful way of prioritising which features to look at for each 'decision step'. Attentive transformers may ask questions about hundreds of different features at each step. It also has long-term memory built in - it remembers the outcome of previous decision steps and the actual data behind the decisions.

The feature transformer looks at all the features assessed and decides which ones are indicative of fraudulent/genuine behavior. The feature transformer has decision-making processes internally built into its architecture.

Architecture limits overfitting

TabNet architecture can prevent overfitting issues which occur in random forest. It does this in two ways: through loss function and through the feature transformer.

We can limit the granularity of the model so so it learns the repetitive features of fraudulence rather than the individual features of a single fraudulent transaction. This makes the model more general so it can make predictions on new data which doesn't look identical to what it has seen before.

The feature transformer emphasises reusable decision-making processes - in other words it 'remembers' how it makes decisions. This means that if the feature transformer sees the same feature data more than once it will try to make a decision in the same way each time - preventing further granularity down the decision-making chain.

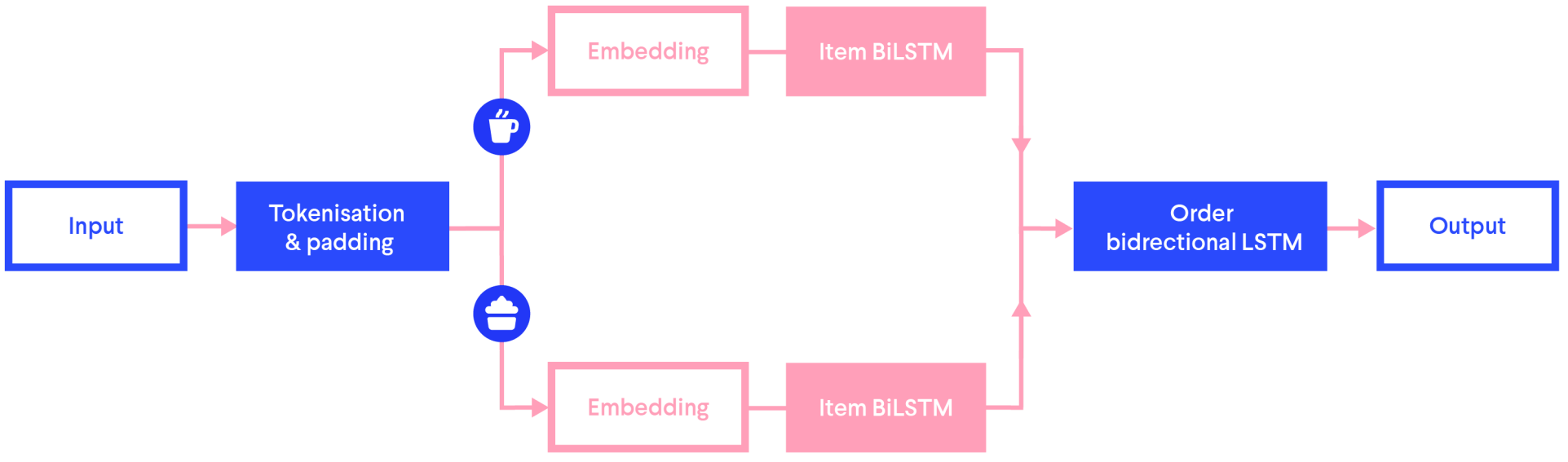
Natural language processing (NLP) micromodel

NLP models are often used in applications such as voice recognition, email spam filters or translation services.

At Ravelin, the NLP model purpose is to assess text features, such as email, order basket content, delivery notes and other text items.

NLP model design

An example NLP architecture for a food delivery order basket could look like this:



Tokenization/padding is how we convert the text into numbers that the model can understand as data. In our model we tokenize at the character level to enhance model flexibility towards new items as well as new features.

This means it’s easier to find repetitions in the data - for example a repetition of a string of random letters in fraudulent email addresses with incremental changes.

Embedding and item BiLSTM - items are individually embedded and passed through a bidirectional long short-term memory (biLSTM) block to encourage the model to learn similar items, eg. latte and cappuccino. At the item level, the LSTM adds extra context to the text, grouping text where appropriate. For example, it may group ‘black’ with ‘coffee’ and ‘tea’ with ‘milk’.

Order BiLSTM - a final order biLSTM learns typical order baskets - eg. food item and a drink. Having a bidirectional LSTM means the chronology of the order doesn’t have an adverse effect eg. it doesn't adversely affect the mode if someone orders drink first then food, or food then a drink. This also promotes learning of orders and other user behaviors.

Advantages of our NLP architecture

NLP can remove the element of human bias which goes into building text features - for example an English-speaker might only be familiar with English text and build features based on their own knowledge and the rules of English language. NLP can encompass many different languages without the human resource overhead and reduce this type of bias.

With conventional text feature extraction, a new feature or new text field requires a person to manually build this feature into the model. NLP will do this automatically, reducing the human burden when introducing new products, email domains etc.

Anomaly detection model

Another architecture we use is a deep autoencoding gaussian mixture model (DAGMM). The model condenses the data and learns a compressed representation, allowing it to confirm new customers as typical of the existing dataset or identify outliers.

At the blue nodes, the model asks questions to compress the data, then on the other side it must ask the questions in reverse so that the data comes out looking the same. The model will try to distill the key signals in the customer information.

The anomaly detection model assimilates all the data and based on the combination of all features, it decides if it should allow or prevent the transaction. We continuously update the anomalous behavior model to ensure it is reflecting the current data.

This model's purpose is to classify outliers and anomalous new user behavior. It asks the question - "Does this new customer fit the typical customer profile from my dataset?"

If the answer is no it doesn't necessarily mean the customer is fraudulent. This is interpreted as a signal which is passed to the classification model (along with all the other model outcomes) and blended, before giving the customer a fraud score.

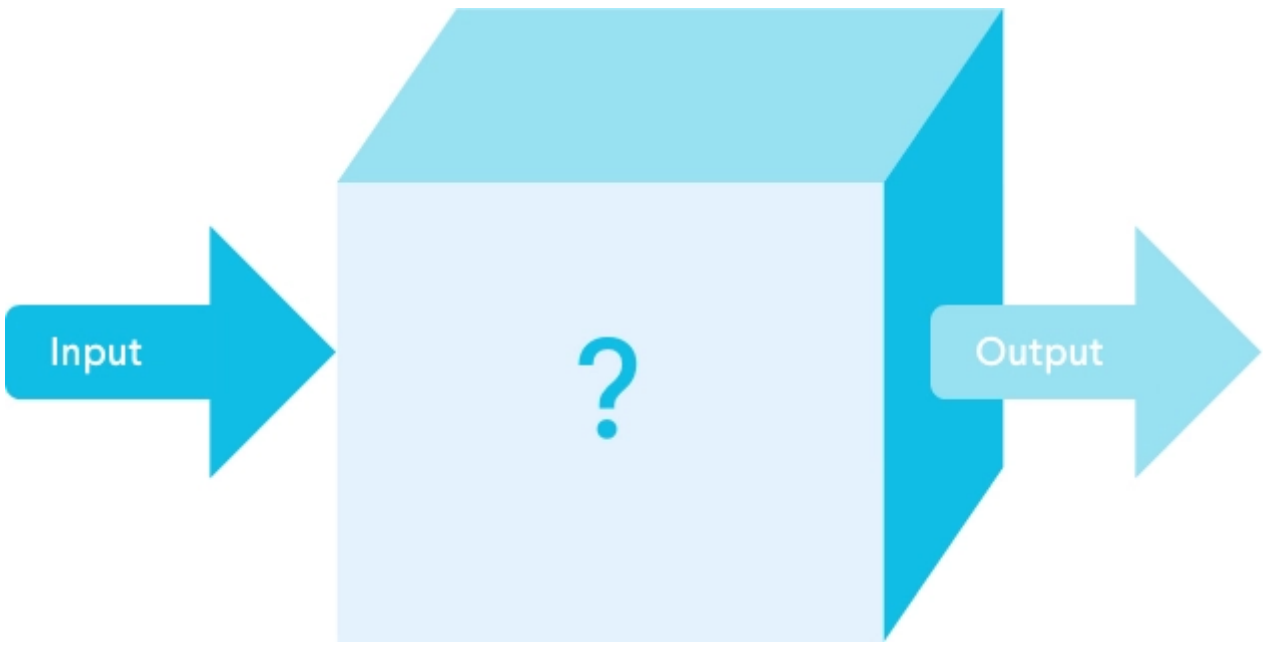
Blending the models

As mentioned above, these are our main three architectures, but we do use other models on top of these.Each model contributes signals to the classification model, and these are blended before producing a final score. One model on its own cannot make a total decision, this is only done by the classification model. The classification model uses a simple neural network structure to assess all the signals from the various models.

Learn more about the machine learning practices and model architectures we use in our guides here or please get in touch to find out more.

Understanding the results - looking inside the black box

Machine learning is often called a black box as you can't really inspect how it's doing what it's doing.



Although it's difficult to inspect everything the model does, we can get an understanding of how it works through testing cause and effect. We make subtle, controlled changes to the data we feed into the model and measure the output so we can tell what data the model favoured when it made the prediction, eg. did it prioritise suspicious email addresses or high transaction values. Every single customer's prediction is instantly explained in full on the dashboard.

How human insight complements machine learning

When used successfully, machine learning removes heavy burden of data analysis from your fraud detection team. The results help the team with investigation, insights and reporting.

Machine learning doesn't replace the fraud analyst team, but gives them the ability to reduce the time spent on manual reviews and data analysis. This means analysts can focus on the most urgent cases and assess alerts faster with more accuracy, and also reduce the number of genuine customers declined.

Machine learning makes the role of a fraud analyst more efficient, as their time is freed up to do more strategic work. Analysts improve and optimise machine learning fraud detection systems through reviewing and labelling customers and tuning the rules. Machines are exceptionally good at doing the heavy lifting in data analysis, number crunching and output. They work

Machines are less good at dealing with uncertainty. There are cases that are new, or that are difficult, or somehow different. Edge cases are those that require more attention and may be difficult to determine - this is where the human insight comes in and provides massive value.

The expert human intervention here is not just at the point approving a transaction. It’s more a case of analysis after the event and labelling the data in a way that gives rapid feedback to a machine. Remember, labelled data is the ultimate training set for a machine. So the more confirmed behaviour labels it can receive the more accurate a result there is likely to be.

During live fraud events, fraud analysts can use the manual reviews to let us know when an attack is happening. Human insight is key to stop fraud attacks and limit the negative impact. Read more about how machine learning is changing the way fraud analysts work in this article.

SOLUTIONS

Fraud solution suite

Accept for payments

INSIGHTS

Fraud and payments survey 2020

Online payment fraud

PSD2 and SCA

Machine learning for fraud

Link analysis

Account takeover

Global payment regulation map

RESOURCES

Blog

Guides

Virtual events

Webinars

API & developer docs

Tech blog

COMPANY

About

Careers

Customers

Contact

Partners

Privacy policy

Ravelin support

Open source licenses