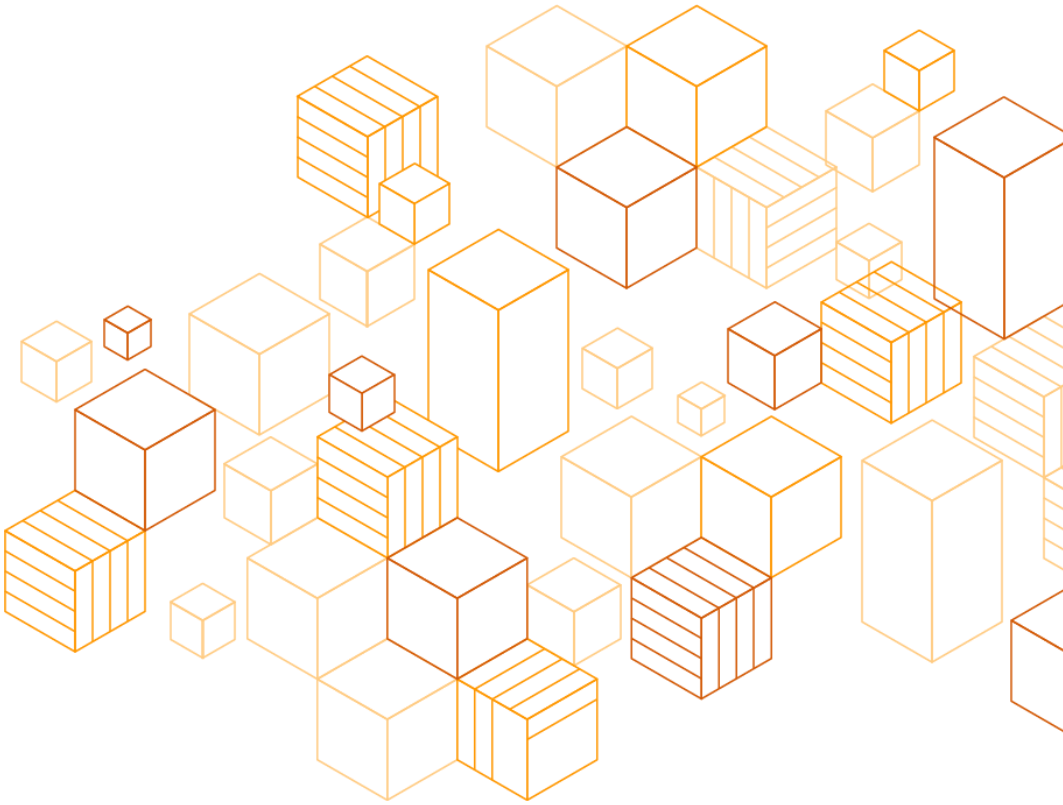
Detecting New Account Fraud and Transaction Fraud with Amazon Fraud Detector

Technical Guide

November, 2021



Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

About this Guide

Companies today use everything from simple spreadsheets to complex software in a bid to detect online fraud. This paper introduces the problem of online fraud, its terminology, challenges, and use cases. This document walks through the steps of detecting online fraud using machine learning, and references how Amazon Fraud Detector powered by an automatic machine learning (AutoML) pipeline can help simplify the process.

Overview

About Online Fraud

Tens of billions of dollars are lost to online fraud each year, and fraud management systems have become critical to helping companies protect their customers, their reputation, and their bottom line. According to Javelin Research, total combined fraud losses climbed to $56 billion in 2020.

Online fraud can take on different forms and can happen at various stages depending on the business. Two common fraud types are New Account Fraud and Transaction Fraud. 1) New account fraud is generally defined as fraud that occurs within the first few months of an account opening. These accounts are opened with the sole purpose to commit fraud. For example, fraudsters use stolen or synthetic identities to open new bank accounts, intending to max out their credit limits before disappearing into thin air, usually within 90 days. Indeed, the FTC estimates that in the US, credit card new account fraud was up 24% year-on-year in 2020. 2) Transaction fraud occurs when bad actors use stolen or compromised payment instruments to make transactions. Transactions can happen by mail or on the phone, but they mainly happen online. According to Forbes, organizations are losing an average of $4.5 million per year due to online fraudulent transactions. This loss is potentially just the tip of the iceberg, especially for larger businesses. There are other common fraud types, e.g., account compromise, however this technical guide will focus on the first two types.

Challenges of a Fraud Management System

Traditional fraud management solutions present a number of challenges. First, companies often must invest in complex, expensive, proprietary hardware and software that can take months to test and deploy, require specialized skills to configure, and need consultants to implement. Second, these systems tend to be inflexible and unable to adapt to companies' changing needs. For example, successful start-ups can quickly outgrow their existing solution's functionality, and must invest in a new solution or spend resources customizing their systems. Large enterprises can experience peak transaction volumes that their existing solutions cannot process. Finally, companies need to make additional investments in hiring ML experts if they want to use machine learning (ML) to catch more fraud or increase automation in their systems. Overall, these challenges lead companies to reject good customers, conduct more costly

manual fraud reviews, and miss opportunities to drive down their fraud loss rate.

Why Amazon Fraud Detector?

Amazon Fraud Detector (AFD) is a fully managed service that provides scalable, easy-to-use, real-time fraud detection. With AFD, businesses of any size can detect fraudulent online activity in milliseconds without investing the time and resources traditionally required to build and maintain a fraud management system. AFD evaluates online activities like account sign ups or online orders using a combination of machine learning (ML) and rules to assess the risk that the activity is fraudulent. Using expertise developed from 20 years of detecting fraud on Amazon.com and AWS, AFD provides customers with tailored-made fraud detection ML models that learn from their historical fraud data and from past attempts to defraud Amazon.

With AFD, customers get a scalable fraud management system without the infrastructure costs and inflexibility that comes with traditional systems. AFD is cloud-based, so it minimizes the amount of infrastructure companies need to manage and reduces the time they spend building, configuring, and maintaining their fraud system. It automatically scales up or down, enabling customers of any size to perform millions of risk evaluations per hour.

Considerations Before Attempting to Solve a Fraud Detection Problem with ML

Determine a Use Case

As mentioned above, this technical guide focuses on two common fraud types: New Account Fraud and Transaction Fraud. In AFD, there are two model types designed for these two use cases.

- New Account Fraud ← Online Fraud Insights (OFI)
- Transaction Fraud ← Transaction Fraud Insights (TFI)

Keep in mind that there are other use cases, e.g., promotion abuse, that can be tackled by OFI or TFI. Because every AFD model is trained on a customer's own data, unlike off-the-shelf solutions, AFD allows for a custom model for very niche scenarios so that customers can get the same benefits of a highly engineered off-the-shelf solution.

AFD is continuously developing new model types to deal with emerging use cases. If customers choose to use ML models, they simply select a model type such as Transaction Fraud Insights (TFI) and upload their historical data. AFD will use this data, along with insights from past attempts to defraud Amazon, to automatically train, validate, and deploy a custom fraud detection ML model.

OFI model type is designed to detect fraud when the entity (e.g., customer or merchant) is only seen once, whereas TFI is designed for business scenarios with returning customers, like online transaction, where events are linked closely by the entities. For a detailed description about OFI and TFI model types, see *Data Aggregations* below.

Define Evaluation Criteria

A typical way of evaluating effectiveness of a fraud detector solution is to look at the True Positive Rate (TPR, also known as Fraud Capture Rate) at a specified low False Positive Rate (FPR). Depending on the use case and rarity of the fraud, evaluating the Fraud Capture Rate (True Positive Rate) based on 1% – 5% False Positive Rate generally works well, especially when combining it with a monetary value of fraud prevented at each threshold. This makes the value of a fraud detection solution, e.g., AFD, immediately tangible for business people to understand.

Table 1 - Example of TPR, FPR for various score thresholds

FPR	TPR	SCORE THRESHOLD	Fraud $ Caught
0.01	0.39	950	$2,284,293[1]
0.02	0.52	900	$2,952,298
0.03	0.57	855	$3,249,521
0.04	0.61	810	$3,462,551
0.05	0.67	765	$3,657,968

For a detailed discussion about metrics for evaluating a ML-based fraud detection model, see *Appendix A: Metrics for Evaluating A ML-based Fraud Detection Model*.

[1] These numbers are hypothetical.

How to Detect Online Fraud with ML

This section presents the steps about how to train, tune and deploy a ML model, which is then served as a prediction API.

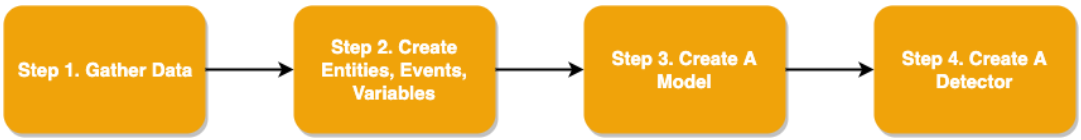


Figure 1 - Steps of building a fraud detector with AFD

Step 1: Gather Data

Machine learning models are only as good as the data used to train them. Data gathering is typically an iterative process of extraction and data profiling to ensure that there is enough training data and enough labels to train a model with. How much is enough? Which variables or features[2] will provide the most meaningful value? This is impossible to quantify rigorously and customers can only rely on rules of thumb. The general rule is, as with most machine learning problems: the more the better. To answer this question more effectively, customers also need trial-and-error.

The AFD team conducted extensive experiments and suggested the following minimum data required for getting a performant ML model.

Minimum Data Requirement

- Online Fraud Insights (OFI)
  - 10k records with at least 400 of those records identified as fraudulent
  - EVENT_TIMESTAMP and EVENT_LABEL are required features (see *Table 2* for descriptions)
- Transaction Fraud Insights (TFI)
  - 10k records with at least 400 of those records identified as fraudulent
  - At least 100 entities with fraudulent records, at least 100 entities with legitimate records

[2] *Variable* and *feature* are used interchangeably in this document.

- EVENT_TIMESTAMP, EVENT_LABEL, EVENT_ID, ENTITY_ID, ENTITY_TYPE and LABEL_TIMESTAMP are required features (see *Table 2* for descriptions)

Table 2 – Descriptions for required features

Features	Description	Requirement
EVENT_TIMESTAMP	The timestamp of when the event occurred. The timestamp must be in ISO 8601 standard in UTC.	Both OFI and TFI
EVENT_LABEL	Classifies the event as fraudulent or legitimate	Both OFI and TFI
EVENT_ID	An identifier for the event. For example, if the event is an online transaction, the EVENT_ID might be the transaction reference number.	TFI
ENTITY_ID	An identifier for the entity performing the event.	TFI
ENTITY_TYPE	The entity that performs the event, such as a merchant or a customer.	TFI
LABEL_TIMESTAMP	The timestamp when the label was last updated.	TFI

As with most machine learning problems, more features generally help. It is recommended to include as many relevant features as possible, such as ip_address and email_address, as AFD performs additional feature engineering on non-mandatory features as well as required features, but it depends on the use case and data availability. While OFI or TFI model template can operate with as little as a few days' worth of events, the general guidance is to collect a minimum of 6 weeks of historic data, though 3 - 6 months of data is preferable. The OFI or TFI model will use EVENT_TIMESTAMP to make sure that the performance metrics reported by AFD is reliable to generalize to future unseen data.

Here is a sample of a dataset for OFI that has the two minimally required features and two more recommended features:

Table 3 - Sample dataset for OFI

EVENT_TIMESTAMP	EVENT_LABEL	ip_address[3]	email_address[4]
2019-10-04 07:02:55	legit	125.164.142.171	fake_millerdavid@example.org
2018-11-27 12:47:03	legit	192.52.200.124	fake_wmichael@example.net
2019-01-08 09:11:14	fraud	169.240.240.125	fake_xkramer@example.org
2018-12-29 19:51:09	legit	192.7.68.168	fake_hnelson@example.com
2019-03-18 08:06:46	fraud	192.52.192.228	fake_andre54@example.com
2019-06-13 05:34:14	legit	169.209.112.148	fake_jcraig@example.com

Data & Label Maturity

Label maturity is an important concept. As part of the data gathering process, customers want to ensure that records have had sufficient time to "mature"; i.e., enough time has passed to ensure that the "non-fraud" and "fraud" records have been correctly identified. It can take 30 - 45 days (or more) to correctly identify fraudulent events, because of this it is important to ensure that the latest events are at least 30 days old or older. For example, in the case of detecting chargeback fraud, it generally takes a statement cycle or more to identify a fraudulent charge. It is generally recommended that that the most recent data in customers' dataset is at least 4 weeks old, but customers know their data best, so they should use their judgment in deciding which data is old enough to be reliable. That said, don't expect that models trained on data from 2018 to perform well when predicting on data from 2020. That is because fraud patterns change over time, sometimes quickly, so the general rule is that the oldest data is no more than a year old. Also, for current TFI models, data aggregations would only be effective for data within a time window where the data is likely to be reliable.

General Recommendations

- Latest training data is at least 4 - 6 weeks old

[3] The IP addresses in this table are fake, only for illustration purpose

[4] The email addresses in this table are fake, only for illustration purpose

- Oldest data is no more than a year old.
- Labels are mature

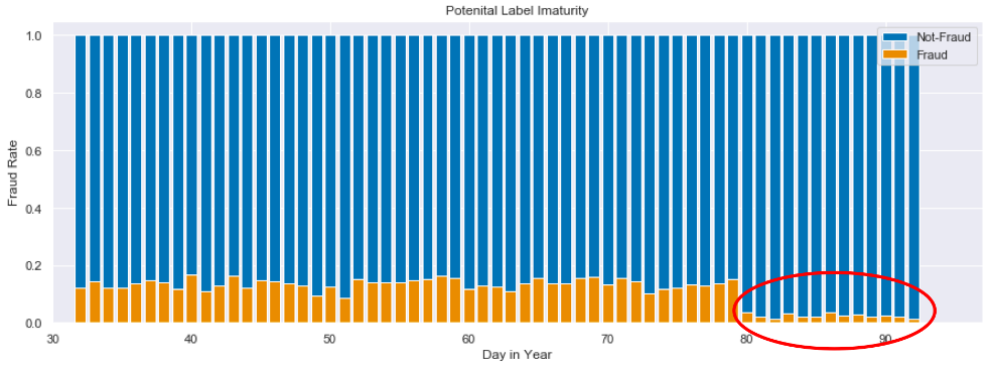


Figure 2 - Example of immature fraud labels

Data Profiling

Profiling historical data typically yields immediate insight into the fraud problem customers are facing. A quick check of the following generally provides an immediate payoff:

- Missing value percentages by column
- Percentage distinct values by column
- Fraud rate over time
- Frequency and percentage fraud by categorical levels
- Histogram of fraud and not fraud for each numeric variable

Customers can use a data profiler which can be found here: https://aws.amazon.com/blogs/machine-learning/train-models-faster-with-an-automated-data-profiler-for-amazon-fraud-detector/

Step 2: Create Entities, Events, Variables

Understand the data model behind the historical data. In the case of detecting online fraud, the concepts of *Entity*, *Event* and *Variable* would naturally come up: an event is a business activity that is evaluated for fraud risk, and an entity is the subject who conducts the business activity. In AFD customers define an “event type” and the “variables” (both names and types) associated with that event type. With AFD, customers generate fraud predictions for events that are instances of the event type.

A variable represents a data element associated with an event that customers want to use in a fraud prediction. Variables can either be sent with an event as part of a fraud prediction or derived like the output of a ML model. AFD has over 30 built-in variable types including IP_ADDRESS, USERAGENT, and EMAIL_ADDRESS as well as 3 custom data types CATEGORICAL, NUMERIC, and FREE_FORM_TEXT. The variable type assignment is the key design choice that enables AFD's machine learning to dynamically reconfigure its data processing based on a user's raw data. Variable type assignments determine how AFD will use the variable as well as what enrichments and transformations are performed on the variable.

Step 3. Create A Model

At the heart of Amazon Fraud Detector is a sophisticated automated machine learning (AutoML) pipeline. It employs the latest advances in machine learning and automation to produce robust and accurate fraud detection models. Below is a description of the key steps in the model development process.

Data Aggregations

Since it was officially launched for general use in 2020, AFD introduced the Online Fraud Insights (OFI) model type which performs well when detecting registration or guest checkout fraud, where events are independent. In business scenarios with returning customers, like online transaction, events are linked more closely by entities (Here each customer is also called an entity). Therefore, it is important to learn from the behavior patterns in the same entity.

AFD released the Transaction Fraud Insights (TFI) model type in October 2021. This model type will automatically generate behavioral features based on an entity's transaction behavior. What makes TFI different is a unique real-time feature store and aggregation engine. Here's how it works: As events are sent to AFD, the feature store and aggregation engine are ingesting these events and generating a historic profile of activity of an entity. The new Transaction Fraud Insights model type detects up to 30% more fraudulent transactions and maintains its performance up to six times longer than Amazon Fraud Detector's previous model type, Online Fraud Insights.

At the time of this writing, no other AutoML solutions in the market offer automated aggregations for their customers. The use of these automated aggregations is an innovation that offers significant benefits over alternatives like SageMaker. For example, if a customer chooses to use SageMaker, it would take time and ML expertise to engineer those features by themselves. And the engineering needed to do accurate

calculations in real time and make them available to a model is not easy for engineers to develop well, deploy and maintain.

Data Validation, Partitioning and Sampling

Once initiating model training, AFD will lift customers' data into memory and perform a number of validation checks. AFD inspects each column for missing values, data type, and checks to ensure that the fraud rate over time is relatively stable. AFD has close to 100 different validation checks built in. Some are column specific. For example, if customers provide an email address, AFD will check that the addresses adhere to a valid-email address format. Some validate data consistency, and others look at missing value rates, for example.

Fraud patterns change quickly over time. Simple out-of-time splitting can underestimate model performance when fraud patterns shift quickly. To address this issue, the AFD's research science team developed a novel train-validation-test splitting strategy to provide more accurate performance estimation on new and unseen data. On the sampling front, it is not uncommon that customers face a severe class imbalance, i.e., low fraud rate. Left un-checked this can lead to poor prediction performance on fraud events. AFD has implemented a novel down-sampling strategy that is triggered when the fraud rate is lower than 5%. This novel sampling strategy improves the model's fraud capture by up to 21% and facilitates faster and more stable model tuning.

Data validation, partitioning and sampling are typically performed by data scientist before training an ML model. Compared with other platforms such as SageMaker, AFD saves data scientists' time by performing all these tasks automatically.

Feature Enrichments

Feature enrichments in AFD will automatically enrich features such as IP addresses. For example, IP addresses are enriched with many additional variables including ISP organization name, IP host name, country, city, state and so on. These feature enrichments are automatically performed on customers' behalf. The enrichments dramatically improve performance of the models and reduce the cost and burden of setting up these common enrichments.

At the time of this writing, these enrichments are not available for free. The cost of enrichments is rolled into AFD. For customers with other platforms such as SageMaker, even if they can obtain these enrichments, they would still need to perform the lookup themselves.

Feature Engineering

AFD performs a number of feature engineering and transformation steps based on the variable types. For example, user agents are parsed and hashed into useable features; free form text fields are processed using a novel text transformation process which has been proven to show more than 10% percent increase in AUC over traditional free-form text hashing; and email addresses are parsed into various pieces and parts, including domains, number of characters, length, uniqueness and more. The AFD research science team is constantly reviewing and testing new transformations. Many feature types, such as free-form text, user agent, categorical variables and others, are transformed automatically before being presented to the model.

AFD's variable-type-specific feature engineering and transformation built in has been the result of a significant amount of research and experimentation, and it is optimized for the fraud domain. This is particularly useful for customers who are not able to invest in data science research. For those customers, a generic solution such as SageMaker is not enough to get them good features and models.

Amazon Embeddings

Amazon embeddings are additional features developed to enrich customers' data based on what Amazon knows about past attempts to defraud Amazon and Amazon subsidiaries. Research shows that Amazon embedding can improve fraud performance in a significant manner for many businesses. For use cases like account registration, these embeddings can mean the difference between stopping fraud at the front door before damage can be done or letting the fraudster in and having to deal with the aftermath of a fraud outbreak.

Amazon embeddings is a unique advantage over other AutoML solutions as it is developed out of fraud data owned by Amazon and its subsidiaries. The AFD research science team is also continuously investigating new Amazon embeddings.

Model Training and Hyper Parameter Optimization (HPO)

In machine learning, hyper parameter optimization or tuning is the problem of choosing a set of optimal hyper parameters for a learning algorithm. A hyper parameter is a parameter whose value is used to control the learning process. AFD trains several different models on its users' behalf using different settings and hyper parameters in order to identify the best performing model. This allows users to zero in on an optimal

set of hyper parameters for a given user's data and use case. Once identified AFD would re-fit the model with the "optimal" parameters and begin its evaluation cycle. Finally, AFD performs one final fit on the full dataset. The purpose is to incorporate all of the data into one final model. Once fit, the model is ready to be added to a detector and deployed to make predictions.

Score Normalization/Calibration

Models produce scores, and the scores are calibrated to false positive rates. Scores are calibrated for two reasons. First in a normal cycle the score distribution can shift after each retraining, which means that business rules that consume scores require frequent adjustment. The second reason is that calibration gives the scores meaning. For example, a score of 950 always results in a 1% false positive rate. The score calibration improves the customers experience, and minimize score distribution shift.

The value that score normalization provides allows customers to train new models and implement them faster without having to re-establish FPR baseline thresholds every time. This saves time and allows for models that learn from new fraud patterns to be more rapidly deployed. At the time of this writing, no other solutions in the market are offering this.

Model Variable Importance

Every AFD ML model will report standard ML metrics such *Area Under Curve* or *AUC*. Interested readers can find more details in *Appendix A: Metrics for Evaluating A ML-based Fraud Detection Model*. Additionally, AFD ML model reports model variable importance values to provide customers with more insight into their model's performance. With model variable importance, AFD now provides a ranked list of model inputs based on their relative importance to the model's performance. This information helps customers better understand their ML models and makes it easier to iteratively improve model performance. For example, if customers observe that their model's performance is driven by IP address, they may choose to include other IP address-related variables in their model.

Step 4: Create a Detector

Now that customers have trained a fraud detector model, they are ready to combine fraud detector models and business rules together into a decision engine. In AFD, this is exactly what a *Detector* does. It is the "Detector" that customers use to make

predictions with. The Detector allows business analysts write IF <condition> THEN <action> style business rules using a simple-to-use interface. Additionally, AFD has native integration with Amazon SageMaker, enabling SageMaker users to deploy SageMaker models within a Detector. *Table 4* below show an example of how to set up rules based on model scores for different business outcomes. Note that *$transaction_model_insightscore* denotes the model score returned by the fraud detector model when evaluating a new event.

Table 4 - Example of rules

Name	Description	Condition(s)	Outcome
rule_1	@1% FPR block	$transaction_model_insightscore >= 950	block
rule_2	Manually review events scoring between 855 and 950	$transaction_model_insightscore >= 855 and $transaction_model_insightscore < 950	review
rule_3	Add additional friction for events scoring between 600 and 855	$transaction_model_insightscore >= 600 and $transaction_model_insightscore < 855	friction
rule_4	Default approval rule events scoring less than 600	$transaction_model_insightscore < 600	approve

Integration of Detector with other AWS services

Integration of AFD with other services is straight forward. Detectors can be called from other AWS services using the get_event_prediction() API. AFD can be integrated with services like Lambda, API Gateway, Glue, Connect, A2I, QuickSight and several others. AFD like other AWS services offers a robust set of APIs enabling customers to automate the creation, deployment and re-training of models and detectors programmatically. Refer to this blog post for an example of AFD's integration with other AWS services.

Comparison with Other AutoML Frameworks

Amazon SageMaker is a fully managed service that allows its users to build custom ML models. On top of SageMaker, Amazon SageMaker AutoPilot is a cloud AutoML service which allows its users to automatically obtain predictions from raw data with a single

API call or just a few clicks. There are other AutoML frameworks in the market, among which AutoGluon and H2O are popular ones. Both use algorithms to automatically infer data types, perform feature engineering based on data types, train a variety of models and stack or ensemble the models.

AutoML saves data scientists from performing tasks such as preprocessing and cleaning data, feature engineering and hyper parameter optimization. Compared with generic frameworks[5] such as Amazon SageMaker AutoPilot, AutoGluon and H2O, AFD's AutoML is unique in the sense that it is optimized for the fraud domain. Each step of the pipeline is specifically designed with the problem domain in mind. In fact, AFD research science team has conducted benchmark of AFD's AutoML against both AutoGluon and H2O on detecting transaction fraud for three different marketplaces of Amazon. It shows that AFD's AutoML outperforms AutoGluon and H2O by up to 12% in AUC.

Table 5 below compares AFD's AutoML with SageMaker, AutoGluon and H2O for each of the techniques listed at the beginning of *Step 3. Create A Model*, as well as some other aspects of AutoML such as stacked ensembles. Note that *customer DIY* means that customers need to do it themselves.

[5] Interested readers can read more about Amazon SageMaker AutoPilot, AutoGluon and H2O.

Table 5 - Comparison of AFD's AutoML with SageMaker, AutoGluon and H2O

	AFD's AutoML	SageMaker	AutoGluon	H2O
Data Aggregations	automatic for TFI	customer DIY	customer DIY	customer DIY
Data Validation, Partitioning and Sampling	automatic	customer DIY	customer DIY	customer DIY
Feature Enrichments	automatic	NA	NA	NA
Feature Engineering	automatic, optimized for fraud domain	automatic with AutoPilot, generic	automatic, generic	automatic, generic
Amazon Embeddings	automatic	NA	NA	NA
Model Training and HPO	automatic	automatic with AutoPilot	automatic	automatic
Score Normalization	automatic	customer DIY	customer DIY	customer DIY
Leaderboard[6]	NA	customer DIY	automatic	automatic
Stacked Ensembles[7]	NA	customer DIY	automatic	automatic

Conclusion

Online fraud is already a billion-dollar business, and it's growing. This technical guide has walked through the general steps of building a fraud detector using ML, together with the use of Amazon Fraud Detector in particular.

Amazon Fraud Detector is the only fully managed service that combines the latest advances in machine learning science, more than 20 years of fraud detection experience, and Amazon's own fraud data. The service makes it easy to identify potentially fraudulent online activities such as fake account creation, transaction fraud and many others. Unlike general purpose machine learning tools such as SageMaker;

[6] It is a table comparing performance metrics for a variety of models.

[7] It is a way of combing different models to boost model performance.

AFD is specifically designed with machine learning techniques optimized for the fraud domain. The service allows customers to develop a powerful and robust machine learning model on their own data within an hour or two of loading their data.

Contributors

Contributors to this document include:

- Roger Li, Applied Scientist, Amazon Fraud Detector
- Shantanu Chandra, Senior Research Science Manager, Amazon Fraud Detector

Document Revisions

Date	Description
November 2021	First publication

Appendix A: Metrics for Evaluating A ML-based Fraud Detection Model

Operating Range

True Positive Rate and False Positive Rate are already mentioned in *Table 1*, the relations between TPR and FPR are called *Operating Range*. Many people may know their fraud rate they often don't know what their false positive rate is and how much legitimate activity they are preventing, so it is important to understand operating range. In AFD, a trained ML model will output a list of True Positive Rate (TPR), False Positive Rate (FPR) as well as Precision corresponding to different *model score* (0-1000 with higher meaning likely to be fraud) thresholds. *Precision* is the percentage of true fraudulent events out of all events detected as fraud. A model score is returned each the deployed model evaluates a new event.

False positive rate (FPR) ▲	True positive rate (TPR) ▽	Precision ▽	Model threshold ▽
0%	44%	96%	995
1%	78%	82%	950
2%	88%	70%	900
3%	91%	63%	855
4%	92%	56%	810
5%	93%	50%	765
6%	94%	47%	735
7%	94%	42%	695
8%	94%	39%	660
9%	94%	37%	630
10%	95%	35%	600
20%	96%	21%	360
30%	97%	15%	215
39%	98%	12%	130
50%	98%	10%	75

Figure 3 - An example of operating range in AFD

With this operating range, customers can balance the False Positive Rate (FPR) and True Positive Rate (TPR) by identifying a *Score Threshold* that matches their business's operating appetite for false positives. For example, in this case, say a customer's business accepts a 3% FPR, this translates into:

- A rule the customer would deploy in a detector (see *Step 4: Create a Detector* about creating a detector): IF **$model_insightscore** >= 855 THEN **"fraud"**.
- Expected "True Positive Rate" or TPR of **91%**, at the score threshold (>= 855).
- Expected "Precision" of **63%**, at the score threshold (>= 855).

ROC Curve

A different way of showing the TPR, FPR for various score thresholds is to draw a *Receiver Operator Characteristic* curve or *ROC* curve. It is simply a different view of score thresholds that highlights the tradeoff between FPR and TPR, where TPR is on

the Y-Axis and FPR is on the X-Axis. By choosing a score threshold customers are essentially determining the expected TPR and FPR of their model.

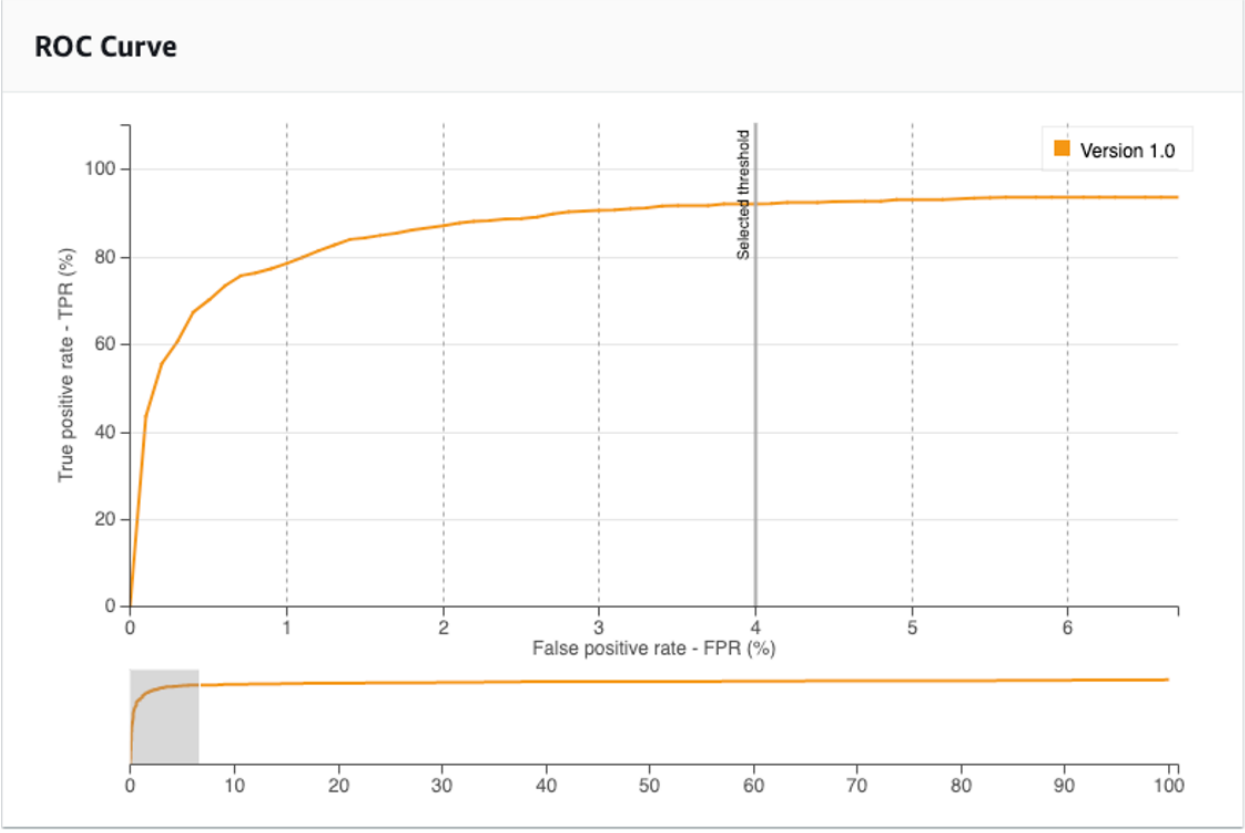


Figure 4 - ROC curve in AFD

Understanding AUC

In ML terms, another common measure that summarizes the model performance in just one number is *Area Under Curve* or *AUC*. It is calculated as the area under the ROC curve shown above. Think of it as a measure of how well the model separates the data into classes, i.e., fraud / not-fraud. It's often mistaken for accuracy which is simply how often a model is right and wrong. A better way think about AUC is to think of it as the percentage chance that the model is able to distinguish between fraud and not-fraud. For example, an AUC of 0.9 essentially means that there is a 90% chance that the model is able to distinguish between fraud and not-fraud. AUC typically ranges between 0.5, essentially a random guess, to 1.0 a model that perfectly distinguishes fraud and legitimate events. As long as the AUC of a model is greater than 0.5, this model is doing better than random chance. That is the model has learned something about how to separate fraud from legitimate events. Of course, a model with a higher AUC is better at

distinguishing between fraud and legitimate events, that doesn't mean a model with say an AUC of 0.7 isn't good it just means that at the same "operating range" it may not be as good at identifying fraud and not-fraud as a model with a higher AUC.

Confusion Matrix

After choosing a score threshold, a *confusion matrix* is a table that describes the expected performance of the trained model. In AFD, the confusion matrix provides an estimate of expected model performance on 100k events based on the selected *score threshold*. Essentially, the score threshold says IF an event's *model score* is GREATER THAN OR EQUAL TO the *score threshold* THEN classify the event as FRAUD, conversely IF it is LESS THAN the *score threshold* it is classified as LEGIMATE. Based on the score threshold this will determine the expected:

- True Positives (TP), i.e., the number of events correctly identified as FRAUD
- False Positives (FP), i.e., the number of events incorrectly classified as FAUD that were actually LEGITIMATE
- True Negatives (TN), i.e., the number of events correctly identified as LEGITIMATE
- False Negatives (FN), i.e., the number of FRAUDS incorrectly classified as LEGITIMATE

Note: simply divide these by 100k and get the expected rates for each classification.

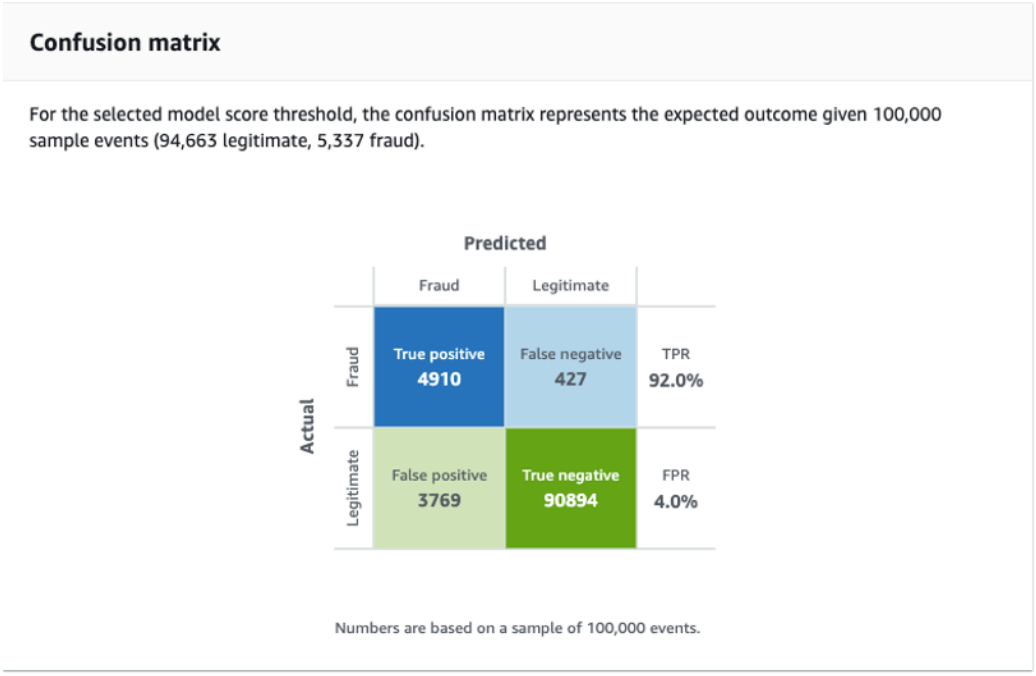


Figure 5 - Confusion matrix in AFD