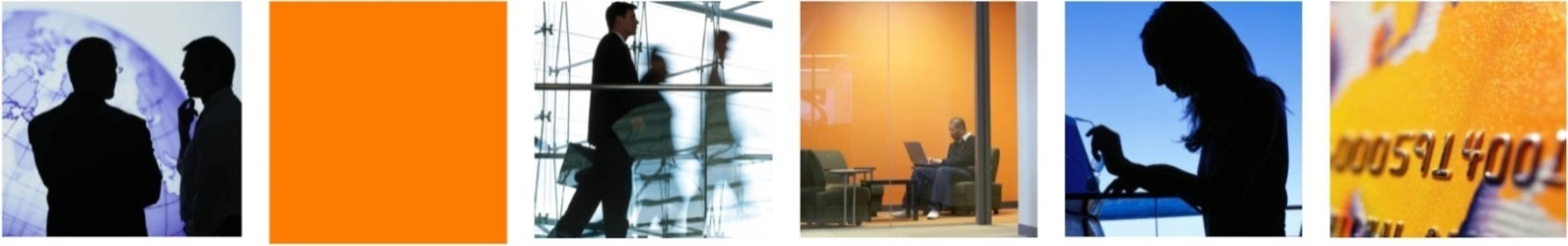


Model Documentation for

Transfer Now®

Navigator ML Model

Version 1

Document Date: 09/21/2022

Contents

[Version Control 3](#_Toc43905684)

[A. Navigator Model 4](#_Toc43905685)

[B. Executive Summary 4](#_Toc43905686)

[C. Model Theory 8](#_Toc43905693)

[D. Modelling Approach 9](#_Toc43905696)

[E. Development Data 11](#_Toc43905700)

[F. Data quality controls 13](#_Toc43905703)

[G. Data Collection – Automated and Manual Transformations 15](#_Toc43905704)

[H. Assumptions and limitations of development data 16](#_Toc43905705)

[I. Modeling Platform 17](#_Toc43905706)

[J. Model Specification 18](#_Toc43905711)

[K. Overall Diagnostic Measures – Goodness of Fit 19](#_Toc43905714)

[L. Aggregate Risk Assessment 20](#_Toc43905715)

[M. Implementation Testing 20](#_Toc43905716)

[N. Sign Off Process 21](#_Toc43905717)

[Appendix A: The Transfer Now® Product 22](#_Toc43905718)

[Appendix B: Performance Statistics 24](#_Toc43905719)

[Kolmogorov-Smirnov (K-S) Statistic 25](#_Toc43905720)

[Kullback-Leibler Information Statistic 25](#_Toc43905721)

[Shannon-Weiner Information Value Statistic 26](#_Toc43905722)

# Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Changes** | **Notes** |
| 1 | 08/15/2022 | Risk Analytics | New model deployed |  |
| 2 | 01/24/2022 | Risk Analytics | LG Boost Estimation Method | Changed reference from XGB to LGB in document |

Navigator Model

|  |  |
| --- | --- |
| ***Model name and version:*** | *TN GEN LGB* Navigator Model v1 |
| ***Model developer*** | Fiserv Electronic Payments |
| ***Model owner*** | Risk Analytics at Fiserv Electronic Payments |
| ***Month/year developed*** | 2022 |
| ***Month/year implemented*** | 2022/Q3 |
| ***Model name*** | *TN GEN Light Gradient Boost Model* |
| ***Approval Date*** |  |

* 1. Executive Summary

## Identification and description of affected business decisions/processes:

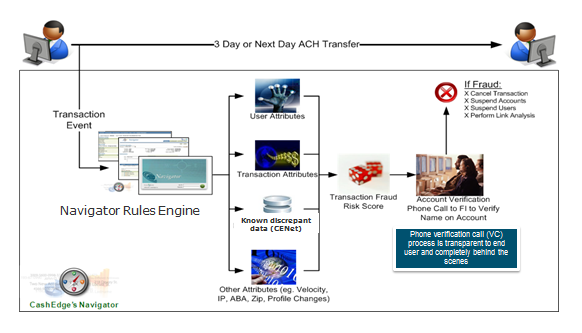
This document provides information about the Navigator fraud model development process, model purpose, application, and model performance. Please note that due to proprietary nature of this model, we cannot disclose all the variables and rules that are a part of the model. We have however provided sufficient details to measure effectiveness and robustness of the model methodology and performance.

## Description of Model Purpose and Application:

The TN model was designed to provide the likelihood (probability) that a third party has gained access to the profile of a legitimate user and is moving money to an account controlled by the fraudster.Based on the fraud probability generated by the model and other risk factors such as the amount of the transfers and Fiserv discrepant data, a priority score ranging from 0-999 is generated and the transfers are routed to a manual account-verification queue.

The priority queues are staffed by a team of specialists who place calls to the non-host banks to verify if the ownership information of the external account matches the credentials provided by the Client Bank. In addition to the ownership verification, the specialist will run link-analysis and extend the review to any other transfers in transit or accounts that are linked to the suspended profiles.

**Diagram1: Risk Mitigation Process**



## Review of Model Development Background:

The revision to the current model will be done if we observe the detection rates drop of see specific patterns of fraud developing for the Client Bank that have not been addressed by the current model. We keep a weekly monitoring process for the existence of returns and deploy fraud-pattern triggers as the information becomes available.

The model was estimated using a Light Gradient Boost ensemble algorithm in Python programming Language. We used LGBMClassifier from lightgbm ensemble libraries in Python to generate Light Gradient Boost algorithm predicted probabilities.

The fraud events that were used as the performance (dependent) variable (“bads”) cover the following cases:

1. Fraud Returns provided by the clients (‘R10’,’R11’,’R05,’R07’,’R29’) through the ACH system
2. **Suspensions-** The user profiles or accounts were suspended by the client or the Fraud Operations department due to confirmed fraud.
3. **Failed Account Verification Calls-** Cases where external financial institutions confirm that end user’s name in the account does not match the name of the user provided by the Client Bank in Single-sign-on and the negative suspension is not lifted by the FI.
4. **Negative Events Reported in Compass.** The clients can flag negative events, fraud suspensions or transfer level actions. The data is collected from the Compass application using the Memo Reason Code with a set of fraud reasons where the client confirms fraud for the transfer.
5. **Negative Events submitted as Files.** Affidavits, negative list flagging emails or phones or accounts can be provided to Fiserv to be populated in the negative databases and applied to the client’s transfers.

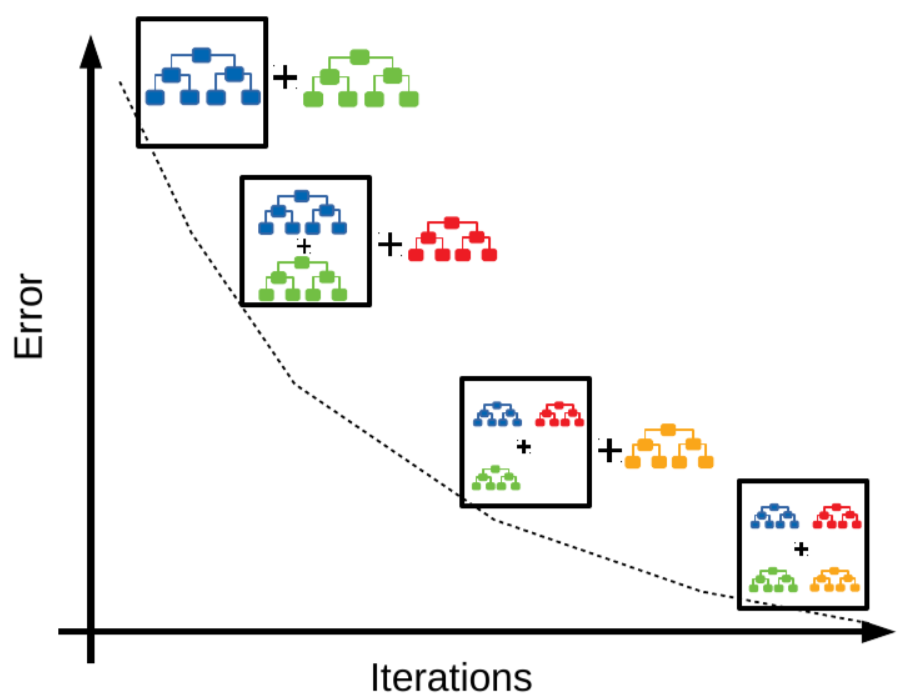
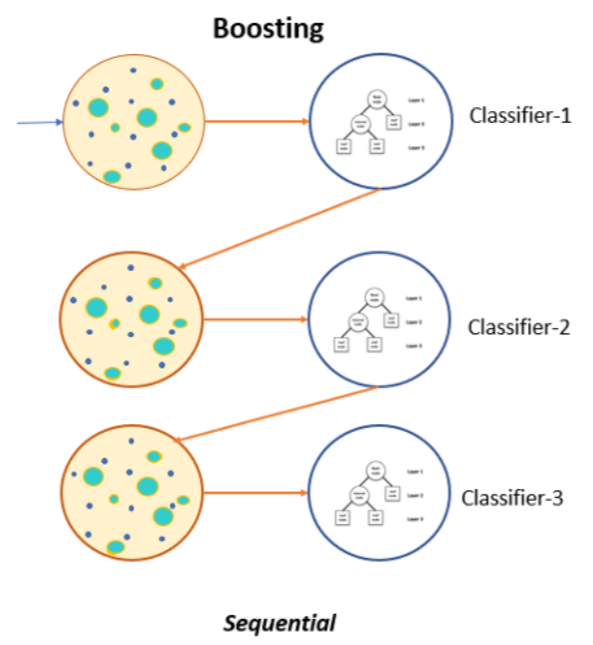
We examined approximately 300 variables that describe user behavior and transaction information. The attributes include factors such as profile and account registration information, the age of the profile, geographical location and distance metrics and other factors regarding the utilization and velocity of transfers.

The chart below shows sample of variables and data source categories used to build a fraud model.

**Figure 1: Risk Factors used as Inputs for the TN Transaction Verification Model \***



For each of the modeling attributes, we performed an exploratory analysis, where we generated the population distributions and the variable correlations to the performance variable and rank ordered the explanatory power. We also calculated weight of evidence and information gain of each attribute to decide which variables should be passed to LG Boost Ensemble algorithm. A Boosting ensemble algorithm combines the decisions from many weak models (Ex: individual decision trees) to improve the overall performance or to form a strong algorithm. Trees are added one at a time, and existing trees in the model are not changed and a gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss. The output of the new tree is then added to the output of the existing sequence of trees to correct or improve the final output of the model. A fixed number of trees are added, or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset. The final policy installed is a combination of ML model generated fraud probabilities and cumulative amounts, and hits in the Fiserv discrepant database (accounts, email or unique phone identifiers of users with confirmed fraud).



## Regulatory mandates/guidance:

This model development process and methodology is consistent with industry best practices for building models for fraud detection.

## Identification of Key Assumptions and Limitations:

**Key Assumptions**

This model is applicable to Transfer-Now® product only

This model relies on completeness and accuracy of data provided by the Client Bank in SSO. It assumes that the information provided by the client in the SSO regarding the identity of the primary account holder is accurate and complete and that the client will report the elements required to track the fraud returns from the ACH system and the suspension information via the Compass® application.

Clients record pertaining to transaction performance in a timely and accurate fashion

Client’s ID authentication and credentialing policy is robust

Model is dependent on accuracy and availability of data provided by third party data sources such as EWS, FIS, Lexis Nexis, etc.

Fraud patterns hold relatively stable for shorter periods.

Account opening policy and authentication of log in credential by the bank is robust

Fraud patterns may change materially in a short period of time. However, such level of specificity relies on information that might vary over time, such as the receiver financial institutions, the velocity of use and connection patterns. It is well known that fraudsters tend to modify their attack approaches and tend to modify their attack patterns, which may result on variation on the stability of the model over time. The rules and priority scores need to be regularly updated to maintain an SLA level contracted by the client.

**Known Limitations**

The model is applicable to Transfer Now® product only.

This model may not detect credit risk or bust-outs.

This model was not designed to prevent third party usage or errors in the usage from the end-users. We have observed in several cases the users transfer to a known-third-party adding the account numbers that do not belong to the end-user but to a third party. This creates events that look just like a take-over events and are treated by Fiserv as such. If the banks decide to remove suspensions and lift the negative flags for the end-user, all these reversed events are treated as non-fraud for the model training. However, this fact reduces the effectiveness of the models to detect account take over appropriately and the detection rates for the clients allowing third-party usage are lower than for the rest of the population.

Fraud patterns may change materially in a short period of time. It is well known that fraudsters tend to modify their attack approaches and tend to modify their attack patterns, which may result on variation on the stability of the model over time. The rules and priority scores need to be regularly updated to maintain losses.

To maintain the current fraud detection rate model has to be continually monitored and updated as the need dictates.

* 1. Model Theory

## LG Boost for Fraud Pattern Detection

Model Theory: Ensemble Models (Ex: Gradient Boosting) for Fraud Modeling. Gradient Boost algorithm is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the model of the classes (classification) or mean prediction (regression) of the individual trees.

Detection of online fraud presents unique challenges that pose some specific constraints in the type of models that can be used among the most important are the following:

**1. Diversity of fraud pattern across the populations.**

Fraud tends to be distributed in small pockets of population, and the patterns change depending on how fraudsters try to penetrate the system.

**2. Low frequency for the dependent variable.** The unit attempted fraud rates vary between 60 bps and 3% on steady state online transfers. Given the low frequency it creates a significant challenge in developing models with low false positive incidence.

**3. High variability in the patterns over time.** The algorithms selected for fraud detection need to be able to deal with variety in patterns and to isolate the groups of population affected for inspection. At the same time, the techniques selected must be able to re-estimate models and deploy them in a short period of time.

**4. High cost of false negatives (misclassification).** The transfer- amount at risk varies between $500 and $1,000,000 dollars. The cost of the false negative represents the losses associated with the fraudulent transfers. The cost of a false positive represents the review of a good customer that may not complete a transfer.

Ensemble models are great tools to achieve a high specificity (high detection) on small pockets of population and fit nicely multiple patterns present without the need to isolate for specific segments. They also provide a higher stability than stand-alone segmentation tree models or logistic regression estimates. They are relatively simple to implement and robust in the face of population changes.



Ensemble models (Ex: Light GBM) are great tools to achieve a high specificity (high detection) on small pockets of population and fit nicely multiple patterns present without the need to isolate for specific segments . They also provide a higher stability than stand-alone segmentation tree models or logistic regression estimates. They are relatively simple to implement and robust in the face of population changes.

* 1. Modelling Approach

## Ensemble Models: Adding the Forecast of Many Trees

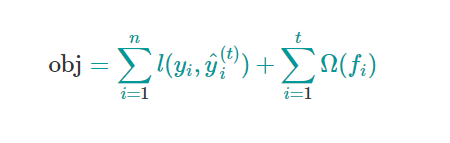
To address some of the limitation regarding the variability of the forecast and adapt the reduction of variance, a common technique is to add or average the result of the multiple trees that are fit on subsets of the modeling sample. . A Boosting ensemble algorithm combines the decisions from many weak models (Ex: individual decision trees) to improve the overall performance or to form a strong algorithm. Trees are added one at a time, and existing trees in the model are not changed and a gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss. The output of the new tree is then added to the output of the existing sequence of trees to correct or improve the final output of the model. A fixed number of trees are added, or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

Part of the problem of constantly adding more variables and increasing the size of trees added into a forecast is the increase of variance, the tendency to overfit patterns that increase the error of the forecast. To prevent the inclusion of spurious information and overfitting, it is necessary to determine the appropriate depth of the branches and whether to further add trees into the weighted forecast.

## Automating the Splits for Many Trees

Now that we introduced the model, let us turn to training: How should we learn the trees? The answer is, as is always for all supervised learning models: Define an objective function and optimize it.

Let the following be the objective function a combination of a loss function and penalty function that will be used to reduce the bias/variance error. The regularization parameters allow us to reduce the unnecessary growth for the trees.

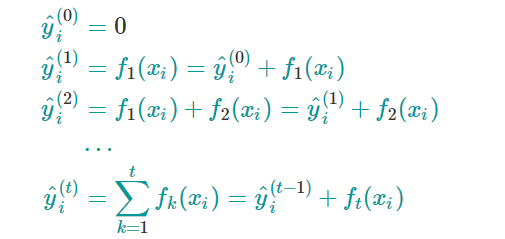


1. **Additive Training:**

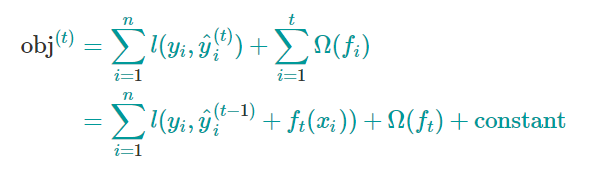
The first question that we want to ask: What are the parameters of trees?

You can find that what we need to learn are those functions fi, each containing the structure of the tree and the leaf scores. Learning tree structure is much harder than traditional optimization problem where you can simply take the gradient. It is intractable to learn all the trees at once. Instead, we use an additive strategy: fix what we have learned and add one new tree at a time.

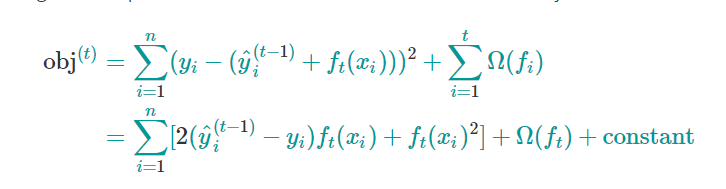
We write the prediction value at step tt as y^(t)I, then we have



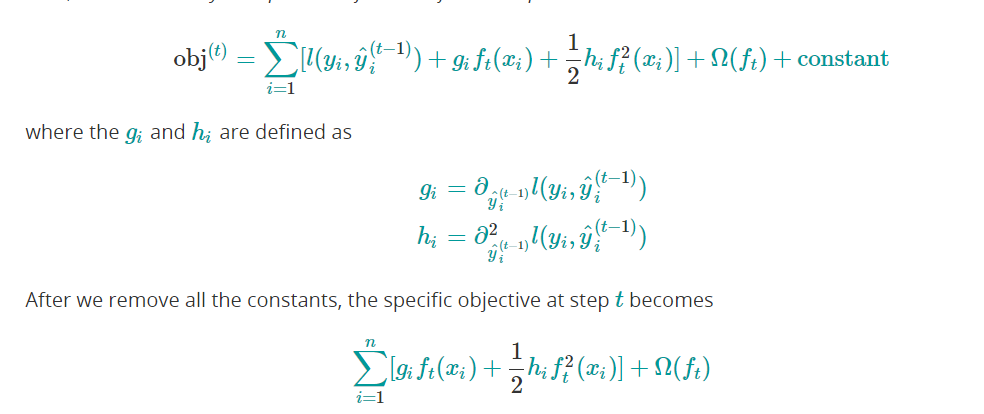
It remains to ask: which tree do we want at each step? A natural thing is to add the one that optimizes our objective.



If we consider using mean squared error (MSE) as our loss function, the objective becomes



The form of MSE is friendly, with a first order term (usually called the residual) and a quadratic term. For other losses of interest (for example, logistic loss), it is not so easy to get such a nice form. So, in the general case, we take the Taylor expansion of the loss function up to the second order***:***

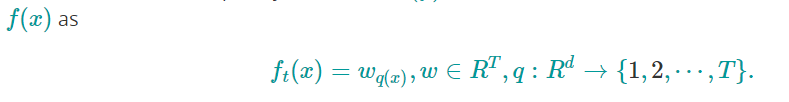


This becomes our optimization goal for the new tree. One important advantage of this definition is that the value of the objective function only depends on gi and hi.

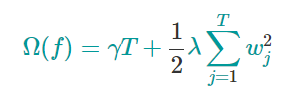
This is how XGBoost supports custom loss functions. We can optimize every loss function, including logistic regression and pairwise ranking, using exactly the same solver that takes gigi and hihi as input!

**2. Model Complexity:**

We have introduced the training step, but wait, there is one important thing, the regularization term. We need to define the complexity of the tree Ω(f). In order to do so, let us first refine the definition of the tree.



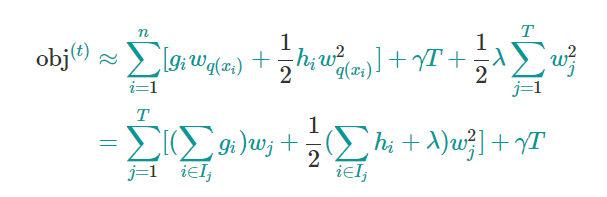
Here w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves. In XGBoost, we define the complexity as



Of course, there is more than one way to define the complexity, but this one works well in practice. The regularization is one part most tree packages treat less carefully, or simply ignore. This was because the traditional treatment of tree learning only emphasized improving impurity, while the complexity control was left to heuristics. By defining it formally, we can get a better idea of what we are learning and obtain models that perform well in the wild.

**The Structure Score:**

Here is the interesting part of the derivation. After re-formulating the tree model, we can write the objective value with the tth tree as:

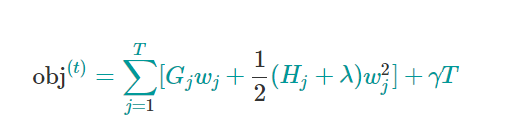


where Ij={i|q(xi)=j} is the set of indices of data points assigned to the j leaf. Notice that in the second line we have changed the index of the summation because all the data points on the same leaf get the same score. We could further compress the expression by defining





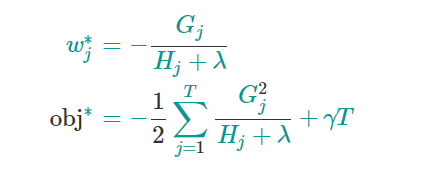
So, the compact form of the objective function becomes dependent on the first and second derivatives with respect to the variables in the model and a penalty term (gamma) for the total size of the tree nodes.



In this equation, wj are independent with respect to each other, the form



is quadratic and the best wj for a given structure q(x)and the best objective reduction we can get is:

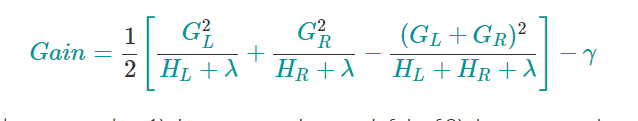


The objective functions can be determined using the first and second derivatives for the loss function with respect to the mapping/splits for the observations to the nodes, and the parameters lambda and gamma, which penalize the total size of the tree (T) and a generic Lagrange penalty form.

In practice, the Lambda term include not only a quadratic measure (L2) penalty, but also the absolute term (L1) that includes the number of variables; these terms allow to reduce the addition of variables that increase bias in the forecast and the inclusion of spurious variables that do not quite contribute to reduce the error of the function.

**Tree Structure and Gains of adding a Partition (Leaf):**

Now that we have a way to measure how good a tree is, ideally we would enumerate all possible trees and pick the best one. In practice this is intractable, so we will try to optimize one level of the tree at a time. Specifically we try to split a leaf into two leaves, and the score it gains is



This formula can be decomposed as 1) the score on the new left leaf 2) the score on the new right leaf 3) The score on the original leaf 4) regularization on the additional leaf.

We can see an important fact here: if the gain is smaller than γ, we would do better not to add that branch. This is exactly the **pruning** techniques in tree based models!

For real valued data, we usually want to search for an optimal split. To efficiently do so, we place all the instances in sorted order, the regular boosting technique will list all the features, bin them, and then use the gains from the split to cut the tree.

XGBoost uses pre-sorted algorithm & histogram-based algorithm for computing the best split.

First, let us understand how pre-sorting splitting works-

* For each node, enumerate over all features
* For each feature, sort the instances by feature value
* Use a linear scan to decide the best split along that feature basis [information gain](https://en.wikipedia.org/wiki/Information_gain_ratio)
* Take the best split solution along all the features

**Structural Differences between LightGBM and XGBoost:**

LightGBM uses a novel technique of Gradient-based One-Side Sampling (**GOSS**) to filter out the data instances for finding a split value while the LightGBM tries to use the fact that the nodes with the higher gradient values require additional splitting and depth of penetration, since they the loss gain is already higher for these nodes.

GOSS keeps all the instances with large gradients and performs random sampling on the instances with small gradients. Gradient represents the slope of the tangent of the loss function, so logically if gradient of data points is large in some sense, these points are important for finding the optimal split point as they have higher error.

The basic assumption taken here is that samples with training instances with small gradients have smaller training error and it is already well-trained. In order to keep the same data distribution, when computing the information gain, GOSS introduces a constant multiplier for the data instances with small gradients. Thus, GOSS achieves a good balance between reducing the number of data instances and keeping the accuracy for learned decision trees.

In simple terms, Histogram-based algorithm splits all the data points for a feature into discrete bins and uses these bins to find the split value of histogram. While, it is efficient than pre-sorted algorithm in training speed which enumerates all possible split points on the pre-sorted feature values, it is still behind GOSS in terms of speed.

The Entropy on a discrete distribution is defined as below.

The entropy measure is maximized (most impure) for the cases where we have an even distribution, with a 50/50 chance of finding a fraud event. Similarly, the entropy reaches minimum when the group splits show perfect concentration of the goods or bads on single membership nodes.

## Comparison to alternative techniques considered.

* Compared to single decision trees, the ensemble model here in our case Gradient Boost classifier has prediction accuracy on par with other known algorithms like gradient descent, boosting, support vector machines or neural networks.
* No feature scaling necessary, and no preprocessing is necessary.
* Easily handles missing values
* Generates more stable forecasts, we see lower degradation between the estimation and validation samples.
* Reduces the amount of work to estimate stable nodes; in the standard estimation for trees, a hold-out sample needs to be kept pruning or validate manually each of the splits in the estimation nodes to ensure the forward forecast for future periods remains stable.

## Assumptions and limitations of chosen technique

* The sampling is representative
* The population shifts are relatively stable during the time that it takes to re-estimate the model.
* Gradient Boost classifier is more robust against sudden changes in Fraud trends, but it is less interpretable.
* Changes in banks underwriting policy may affect the model performance
* It is harder to explain the reasons that the trees are triggering, some other more basic methods, like linear regression or single trees are relatively easy to explain, since for each transfer the triggering condition or weights can be displayed in forms that are easier to consume.
  1. Development Data

Data fields, descriptions, and sources

The model was fit using the Inbound and Outbound transfer population currently flowing through the TN product across the network. The development sample includes 70% of the transfers from June 2021 - April 2022 and validating the forecast on hold out dataset that includes 30% of the remaining data sample.

* Train Dataset: 70% of the Total Population
* Test dataset: 30% of the total Population



The performance variable includes the negative information flowing from the bank review process: – ACH returns, card chargebacks, fraud suspension in Fiserv back-office tool (Compass), Memo Reason Code (decisions by the FI posted in Compass ®, mule and affidavit files provided by the client. Failed account verification calls, resulting on a suspension that is not reversed by the Client FI after review (registered as a fraud suspension in the system).

**Predictive Variables** As part of the authentication and risk-mitigation function, Fiserv collects information from internal and external data sources. Some of the data sources are below:

1. **Bank provided data:** 
   1. User profile information - Name, Address, DOB, Ip-address (available inconsistently)
   2. Account information – Routing number, Account number, Current balance, Date account opened
   3. Fraud data – ACH returns, card chargebacks, fraud suspensions in Fiserv back office tool (Compass), decisions by the FI posted in Compass ® (Memo Reason Codes)
   4. Confirmed Fraud lists provided by the client: affidavits, negative data for emails, phones, account and other contact information.
2. **Third Party data:**
   1. IP geolocation data from Quova, including location of the connection, connection type, use of anonymizers, use of mobile networks, distance to the physical residence, country
   2. Account verification information from EWS/ FIS
   3. Results of the registration attempts using Real-time or Trial deposit registration
   4. Authentication information from LexisNexis
3. **Transfer stats:**
   1. Transfer history
   2. External accounts statistics and party contact information (ownership, status of the account, closure,etc)
   3. Returns information
4. **Fiserv network data:**
   1. Age of Profile
   2. Discrepant data from CENET
   3. Results of link-analysis verification
   4. Link analysis
   5. Manual account verification results
   6. Suspensions and negative flags posted for the same user in other products hosted by Fiserv (BillPay, Zelle, Send Money, Verify Now)

**Data Fields:** The data fields used to build models are proprietary. Information from the registration process, pass/fail status of automated registration, frequency of transfer, geolocation distance between submission and user location and utilization metrics are variables considered as inputs to the models. The variables were collected using a direct replica of the fields available in the production system scoring.

A sample of variables used for Transfer Now® model is given below;

|  |  |  |
| --- | --- | --- |
| Data field | Description | Data source |
| Cum\_txn\_amt\_sent\_3dys | Cumulative transaction amount sent by the user in 3 days | Internal |
| Cum\_num\_txn\_sent\_3dys | Cumulative number of transactions sent by the user in 3 days | Internal |
| Cum\_amt\_rec\_tkn\_1dy\_mprof | Cumulative amount received by the user from multiple profiles in 3 days | Internal |

## Transformations/treatments of data

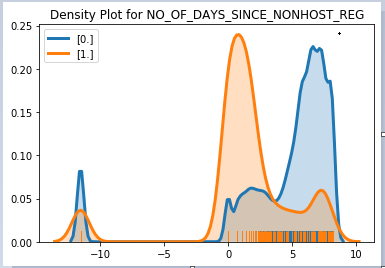
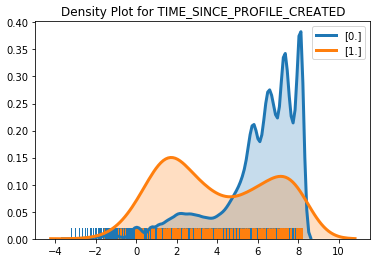
As part of the variable collection, any missing data or attributes are identified from the database and are provided into the algorithm as ID tags with -999 values for the numeric, level and text inputs. The algorithm uses the coded values for missing values and groups them for good/bad concentration and split evaluation.

* 1. Data quality controls

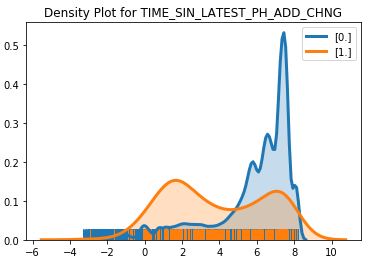
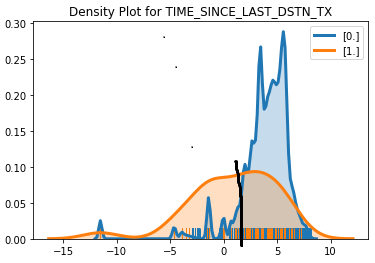
For each of the numeric variables we generated histograms to review the population distribution and to identify irregularities in the data. For factor and level type variables, we created frequency distributions and fraud rates, including segments where missing information is present.

**Density Charts.** For the continuous variables, we created density distributions using the *seaborn* library in Python, for describing the relative distribution of goods vs bads against the predictive variables.

In the chart below we can see how the fraud variable (Target =1) is concentrated on the first 1-20 days since the creation of the profile. The relative concentration shows how roughly 60% of all the fraud events could be captured by reviewing transfers that were submitted in the first 20 days of the creation of the profile and Non-Host account. It is clear, however that there would be a considerable amount of false positives, since roughly 10-20% of the good population volume is posting at the same time, however it would seem natural to segment or partition the population and examine in detail the first 20 days to concentrate review.



The second chart corresponds to the time since the prior transaction posted. Fraudsters tend to post successive transfers until they exhaust the balances in the account and post at a higher rate than the rest of the population.

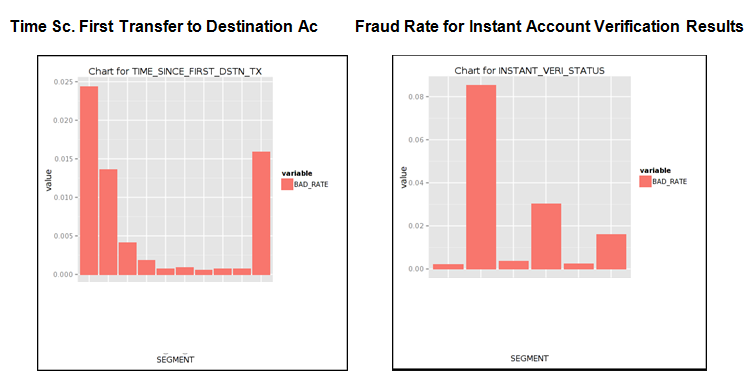


The registration of new information, such as phone or emails also has some predictive value, however some of the changes in the profile confound and correspond to the age of the profile opening, when the phones are initially registered.

**Frequency Charts for Factors and Bin-Based Fraud Rate Charts**

For continuous variables, we split the population in 10 groups over the distribution of the dependent variable. Once the cut-off points for the predictive variable are obtained, the bad rates per group are generated and the concentration of good and bad events on each segment is calculated. The missing values are reflected as N/A as a stand-alone group and will be taken in consideration for the tree calculation splits.

In the example below, we can see the fraud rates for ranges of the Time since the First Transfers to the Destination Account, which is a variable that is available only for the outbound transfers. This is included as a N/A segment at the end of the distribution and the corresponding fraud rate reflects the baseline for inbound fraud.



In the case of discrete variables like the Account Registration type below, we can easily distinguish how the approvals done through the Instant verification matches have relatively low fraud compared to the cases where only partial information was found and the registration was deemed as inconclusive.

The corresponding fraud is lower than in the cases where the information was not available (N/A) and instant verification is not performed. The distributions below show the relative risk of different registration methods. The bad distribution is concentrated in the accounts that were not found during Instant account-registration and went directly into Trial Deposit (between 40 and 60% of the bad population is concentrated there). At the same time we observe a baseline fraud rate for these accounts that is 3 times higher than approval via Real Time or Instant authentication.



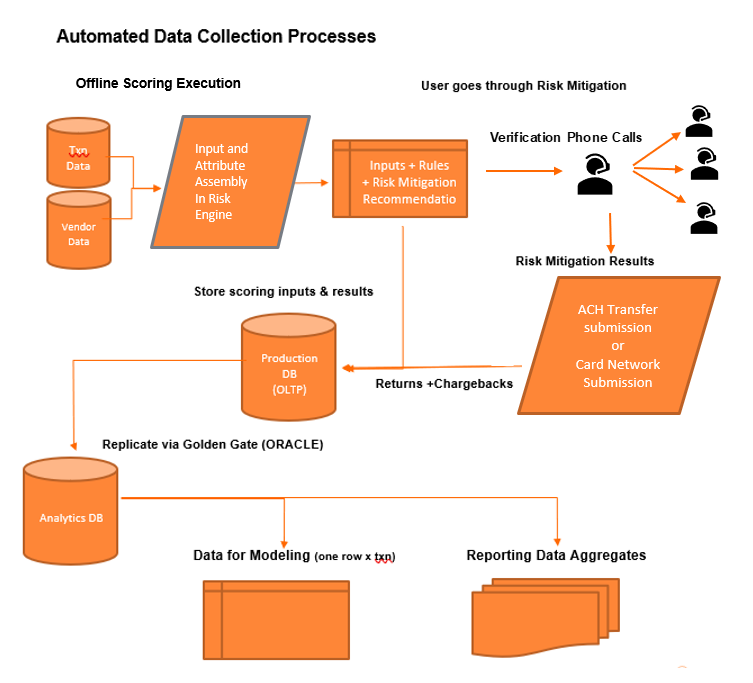
* 1. Data Collection – Automated and Manual Transformations

**Automated Data Collection Processes**

The variables used for model building are a copy of the automated queries running in Navigator that retrieve and generate the inputs from the OLTP production tables. After each observation is scored, the values are stored in the transaction tables. The data is replicated into the Risk Analytics database, where the information is stored within milliseconds of the production process. For each observation, a table with all the modeling inputs for scoring, the risk-scores, the fraud-probability and the results of authentication are stored for tracking and analysis purposes.

The performance definition will collect inputs from the ACH interface table that tracks the submission and returns from the ACH system. The inputs for electronic transfer tables, account verification tables, manual account verification results and the results for the API calls to EWS, QUOVA and FIS are retrieved in real-time (sub-second response times) and provided for scoring.

The inputs from the suspension data and account verification for fraud events are processed manually in the Compass application, which submits the records into our databases. The fraud definition relies on the suspension information codes and results for the account verification that are recorded through the Compass application. The reporting for the negative list is submitted by uploading confirmed fraud cases into the negative databases (CENET) and FraudNet.



* 1. Assumptions and limitations of development data

The model development data relied on the information available for other clients in the network. The implicit assumption is that the registration rules and underwriting provide a similar risk profile for the Client Bank accounts. When we performed the exploratory analysis for Client Bank and compared against similar clients in the network, we observed a set of distributions that are comparable for utilization, average balance.

**Dependent variable**. - The fraud definition relies on the suspension information codes and results for the account verification that are recorded through the Compass application.

The results of the account verification calls are posting into the Compass and Navigator Risk Engine. Failure to verify the ownership of the account results in a suspension that is posted by Fiserv into the profile and account. If the FI does not remove the suspension and the user does not remove it, the information is considered as part of an account take-over attempt and added into the negative definition for modeling.

We will user the debit return information for fraud events submitted in ACH according to the NACHA network to flag fraud. R10, R11, R05, R29 and R07 are considered fraud codes that are being provided by the banks as fraud events.

Discrepant data (negative list) is updated by uploading confirmed fraud cases into Fiserv discrepant databases CENET and FraudNet.

* 1. Modeling Platform



## Modeling platform

The model was developed using H2O, Python scripts and the code developed in Spark, Scikit-Learn platforms for random forests and lightgbm for boosting algorithms, frame processing and data import and export. The final decision rules are created by using ML Model generated fraud probabilities, cumulative amounts, negative list events, high-balance and utilization triggers, non-host target banks and other risk factors. Based on the probability and exposure obtained from the model, we derived the strategy for model implementation in the Risk Engine.

## Description of model code

The modeling objects from H2O and Scikit-learn are transformed into a JSON format using proprietary code. We developed the library to collect the fraud probability generated from each of the end nodes for each of the trees, create an average and use it as the predictive value for the fraud probability of a specific transaction. The model is deployed using a JSON script in the risk engine where at scoring time the attributes are collected in real-time and the code executed as Java package.

The Navigator Risk Engine ® is an in-house platform used to choose the risk mitigation action to be executed. The Transfer Now product uses the full list of input attributes including the fraud probability as inputs to the risk mitigation decision. The risk engine uses the inputs developed in the SQL code and executes the JSON code engine, they are stored as a strategy for change control and review.



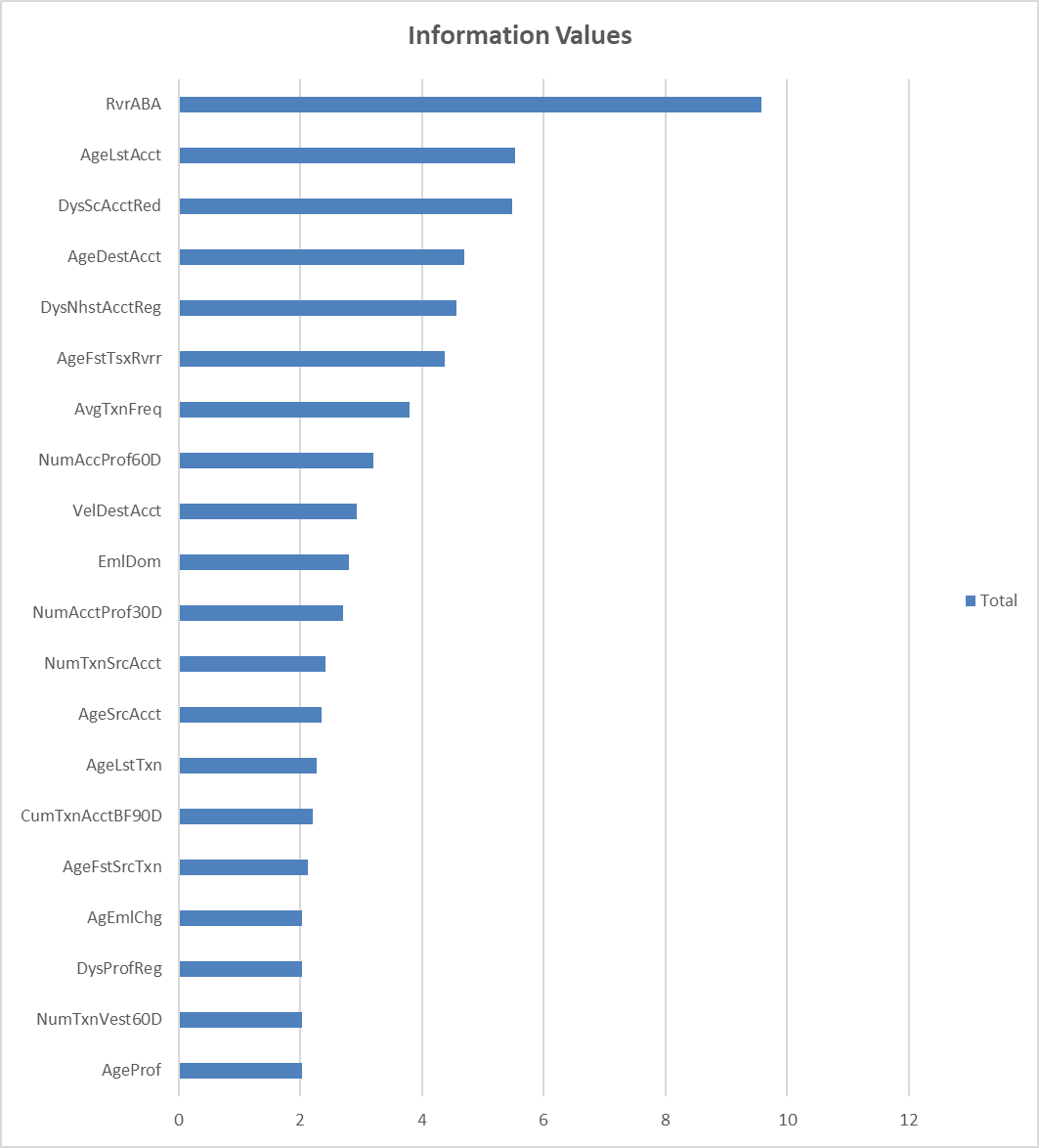
* 1. Model Specification

We used 70% of the transfers posting between June 2021 to April 2022 as the development sample and the remaining 30% of the transfers as the hold out sample.

## Method of choosing inputs to model.

## We calculated weight of evidence and information value of each attribute to decide which variable should be passed to an Ensemble model. Below is the list of some of the variables which has been passed to the model.

**Chart#8: Weight of Evidence (Shannon Information Value)**



We see the weight of evidence providing good separation value for Age of the accounts and profiles, Email Domains and Velocity Variables.

**Data Fields:** The data fields used to build models are proprietary. Information from the registration process, pass/fail status of automated registration, frequency of transfer, geolocation distance between submission and user location and utilization metrics are variables considered as inputs to the models. The variables were collected using a direct replica of the fields available in the production system scoring

The ML algorithm consists of many uncorrelated individual decision trees that operate as an ensemble, performs recursive splits on the randomly selected group of explanatory variables for each tree. Each tree in the ML model outputs a vector representing the predicted probability of each class. The final output is the average of these predicted probability vectors across trees.

* 1. Overall Diagnostic Measures – Goodness of Fit

The gains charts for the routed population used as baseline to the model have been provided below. The gains charts are created by sorting the predictor score/variable from high-priority 🡪 to low priority and aggregating the cumulative percentage of observations in the horizontal axis and the cumulative percent of the target variable in the vertical axis. In the below char, the red line represents the model performance on trained dataset and the blue line represents the model performance on validation dataset.

* 89.88% of the potential fraud dollars are flagged by routing 1% of the total volume.

**Chart #9: Cum % of $Fraud vs. % of Transfers Reviewed**

Chart

Description automatically generated

The curve plots reflect the total % of the fraud dollars that is being captured given a penetration in the review population. Good models tend to have very high cumulative target capture with low percentage of population routed for review.

* 1. Aggregate Risk Assessment

Fiserv will be providing quarterly performance reports on the rank-ordering of fraud incidences by quintile. Performance results will be assessed during each quarterly review.

The above charts are based on the performance for the baseline population selected for benchmarking. The cumulative gains charts show the fraud-detection performance flagging 90% of all amount associated with bad transfers by doing review of the highest risk quantile of the population selected by client, normally ranging between 1% and 3% of the population.

* 1. Implementation Testing

The testing for the implementation is executed in a draft mode, where a sample of observations with known fraud probability from the model design is run through the loaded model in the Navigator Risk Engine. The results are compared against the original forecast provided by the original model and the fit is matched against the results coming from the Navigator Risk Engine.

* 1. Sign Off Process

# Appendix A: The Transfer Now® Product

The TN offline model is part of the risk mitigation policies that are executed during the ACH risk electronic transfer process. In the following paragraphs we cover the areas that are relevant for risk-assessment at transfer time and become inputs to the scoring model.

The risk mitigation process covers 5 check-points in the process for money-movement.

1. Account registration
2. Limit verification controls
3. In-transit risk assessment and mitigation
4. ACH return processing and automated suspensions
5. Fraud investigations and loss recovery

The TN Offline model is part of the in-transit risk-assessment and uses account registration and limit verification information, so we will cover these processes briefly to clarify some of the information that is used as inputs for the model.

**Account Registration Process**

To be able to transfer funds outbound to other financial institutions or inbound from other banks, the user registers the banking information for the external accounts. Fiserv uses a 3 step process[[1]](#footnote-1) for external account registration:

1. Instant account verification against partner databases
2. Real-time login to external institutions and data collection
3. Trial Deposit verification

The results of the account registration, including the failures to match, the mismatches in names, addresses, the number of attempts to register accounts and the time since the last registration attempts become important attributes that are utilized as inputs to the risk-assessment at transfer time. It is well known that accounts with mismatches in information that are approved through Trial Deposit present 4+ times more risk that those approved through the instant verification methods. In the cases where the information does not match the data found in the instant verification, there might be up to a 30% chance that the transfers results in confirmed fraud.

**Limit Verification Controls**

Prior to scoring, the transfer amount is aggregated and verified against the exposure limits set by the client. The information for the utilization and the number of transfers is included in the scoring process, since we have observed that high utilization is correlated also with fraud-risk.

**In Transit Scoring and Risk Mitigation**

Once the limit information has been validated, the transaction is submitted into the Navigator Risk Scoring Engine.

The system will collect user profile information, such as registration date, time of the last login, cumulative amounts transferred up to date and other scoring attributes. It will also perform a match-assessment against the internal databases in Fiserv to check for name and account matches in previous verification and known fraud cases or for positive identification.

The scoring application executes in real-time for the transfer in question and returns a fraud-probability value and a priority score and are queued for risk-mitigation.

**In-Transit Risk Mitigation Process**

The Risk Operations Department will verify that the name on the account matches the credentials provided by the Client Bank for the primary account-holder. If a mismatch is identified, the specialist will document the case and suspend the account and profile and cancel the transfers submitted by the user.

# Appendix B: Performance Statistics

There are numerous statistics available for measuring the differences between two probability distributions, For example, the two populations being examined can be "good" transfers versus "bad" transfers.

Suppose we have a sample of nG accounts from the "good" populations and nB accounts from the "bad" population, giving a total of N = nG + nB observations. For the two populations, there are usually a variety of variables that have been measured for each observations. One variable might be denoted x, where x1, x2… xN are different values of variable x across the N observations.

In general, any variable x can be either of these types:

* A continuous variable taking values that are real numbers. Examples of continuous variables include amount, age of an account, etc.
* A discrete variable has a finite number of categories and takes values that indicate different categories. Examples of discrete variables include account registration method (e.g., instant, real-time, trial deposit), and matches to other profile (e.g., yes or no). .

For purposes of discussion, we'll assume that if x is a discrete variable, then it has J possible categories or levels. For any variable x, there is a probability distribution for the "good" population and a probability distribution for the "bad" population. Let fG (x) represent the probability density function (i.e., PDF) of the "good" population at a given value of x, and fB (x) represent the probability density function of the "bad" population at a given value of x. Essentially, each of these functions indicates how frequently x occurs at a given value. For example, fG (x=10) would indicate how frequently the value x=10 occurs (i.e., the "density" at that value x=10).

For each probability density function, there is also a corresponding cumulative distribution function (i.e., CDF), which is basically a summation of the probability density over all values less than or equal to a given value of x. For x, a discrete variable, the CDFs of the "good" population is:

* 1

The CDF of the “bad” population is:

* 2

For x, a continuous variable, the respective CDFs are:

* 3

* 4

For a given variable x, we want to examine how different are the distributions for the "good" and "bad" populations. This involves trying to somehow measure the distance between fG (x) and fB (x) {or between FG (x) and FB (x)} across all possible values of x. For a given variable x, the distance may be large (i.e., the distributions are much different) or small (i.e., the distributions are almost identical).

By examining distances between the "good" and "bad" probability distributions for different variables, it is possible to determine those variables that provide the "greatest" or "strongest" distances between the two distributions. These strong variables are good candidates for inclusion in a risk scoring model. The following sections describe some of the most common statistics for distinguishing between two distributions (e.g., populations of "goods" and "bads").

## Kolmogorov-Smirnov (K-S) Statistic

The Kolmogorov-Smirnov D-statistic is normally only calculated for a continuous variable x, although there are versions for discrete variables, where the categories (discrete levels) are ranked. It is based on the absolute value of the distance between the CDFs, and is defined as:

* 5

Essentially, for some value of x, the absolute distance reaches a maximum, and the K-S statistic is this maximum distance. While useful, its greatest limitation is that it does not provide a general measure as to how the distributions vary across all values of x.

## Kullback-Leibler Information Statistic

More general statistics can be developed by calculating some distance measure for each value of x and by summing these distances across all possible values of x. One of the earliest was developed in 1925 by R.A. Fisher (a famous statistician), who used a calculation based on the natural logarithm of the ratio of fG (x) to fB (x), namely, ln( fG (x) / fB (x)). Additional variables were later developed, including one originally proposed by Shannon and Wiener, and modified by Kullback and Leibler. The Kullback-Leibler Information Statistic is found in many books on mathematical statistics.

For a discrete variable x, the Kullback-Leibler Information statistic:

* 6

For a continuous variable x, the K-L Information statistic is:

* 7

The K-L statistic provides a very good measure as to the general differences between two distributions across all different values of x, not just at the point of the maximum distance between the two. It can take values of zero or greater, where the greater the K-L statistic, the greater the overall distance between the distributions. Modified versions of this statistic are also used in performing likelihood ratio tests of hypotheses, one of the most common statistical testing procedures in use.

## Shannon-Weiner Information Value Statistic

One limitation of the K-L statistic is that it can potentially take large values greater than zero, and so is somewhat more difficult to interpret. A modified version of K-L was developed to provide an index that is easier to interpret. It is generally known as simply the Information Value (or, sometimes, as "two-sided" K-L Information), and takes values greater than zero but typically not much greater than one. For a discrete variable x, the Information Value is defined as:

* 8

For a continuous variable x, it is defined as:

1. The instant account verification matches the account numbers and primary owner information against the account and ownership data stored in the FIS and EWS systems. In the cases where the information cannot be located, the user is prompted to provide the login and password information for the account. Our systems will log-in in real-time and collect the information from the bank website. If the user declines to provide login information OR the account information cannot be matched, the user is provided with a third option for registration via Trial Deposit. The account will be made available for transfer processing after the deposit information is confirmed. [↑](#footnote-ref-1)