Paper SAS2007-2018

Machine Learning Bots for Fraud, Waste and Abuse

Michael Ames, SAS Institute Inc.

ABSTRACT

One of the most exciting developments has been the advent of machine learning “bots” for combating fraud, waste, and abuse of data. SAS has developed an adaptive intelligent agent, also known as “the bot”, to identify suspicious, anomalous, and fraudulent data across a number of domains. Through continuous monitoring and learning, the intelligent agent has been deployed to augment existing surveillance strategies, identify areas that have been under-surveilled, as well as uncover new and emerging trends.

INTRODUCTION

One of the most often sited uses of machine learning is to combat fraud, waste, and abuse. The challenge is that the typical process is lengthy, requires some amount of domain knowledge and mounds of data. A typical data scientists then has to prepare, explore, transform the data; build and evaluate models, finally deploy the model and test its performance in the “real-world” then evaluate it again, fine tune it and finally if they are satisfied they’ll have to monitor it, and periodically retrain it. All of this work generally takes months to perform, wouldn’t it be more prudent to figure out a way to automate this process?

SAS has developed an adaptive learning intelligent agent, which automates the process of training and deploying a machine learning model to combat fraud, waste, and abuse. The challenge with today’s process is threefold. First it takes an expert data scientist to develop either supervised or unsupervised machine learning models to identify rare and anomalous fraudulent events. Generally a team, of data scientists spend 3-6 months or more to develop a model. Ideally they document the model and explain what it is doing but often don’t. You get a black-box with little or no instructions on what it does. The second problem is the lag between model training, validation and deployment is often measured in months. By the time the model is deployed it’s already going stale. Finally, fraudsters are crafty, they regularly change their behavior and schemes. Between the time that the fraudsters have changed their behavior and your model is retrained, they’ve walked off with the kitchen sink.

INTELLIGENT AGENTS

An intelligent agent is an autonomous agent that is capable of making decisions based on its experience and can learn from the outcomes of those decisions. Unlike supervised learning, learning from labeled data, intelligent agents learn to navigate and adapt to unknown terrain.

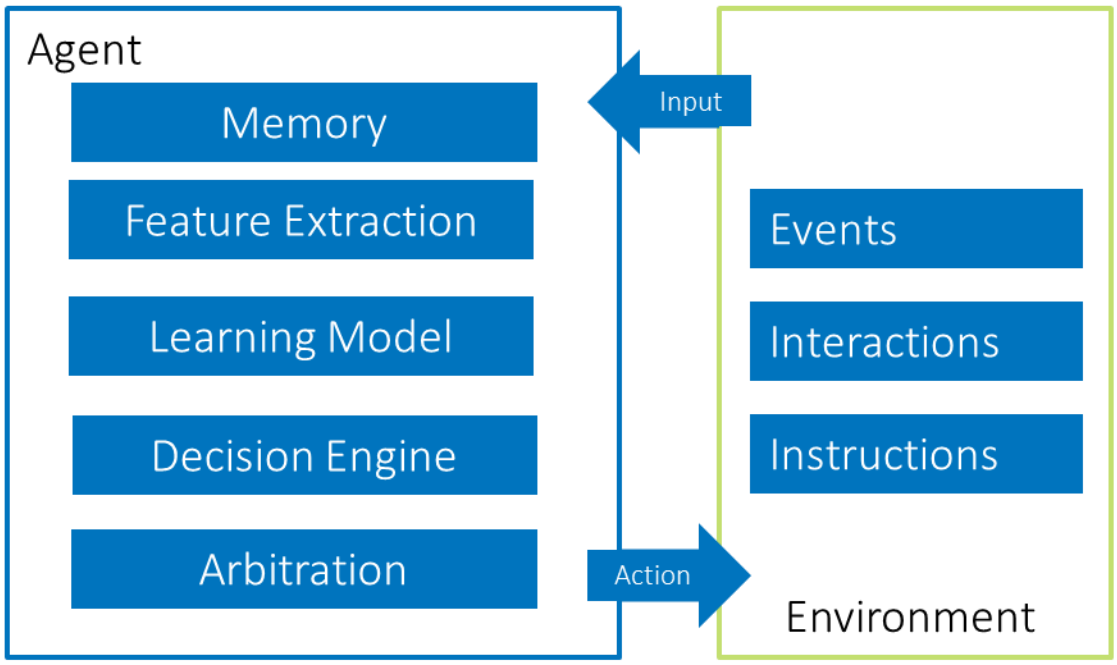

Figure 1. Intelligent Agent.

As figure 1 shows the agent is a process, that contains recent memory and perform things like feature extraction, it has a learning model, a decision engine and performs arbitration between conflicting decisions.

MEMORY

Intelligent agents require some amount of working memory. Just like a human, the agent retains recent events in “working memory”, so that it can adapt to new and emerging interactions with the environment. Typically, this means storing the last several “events of interest” that have been fed into the agent. This might be recent fraud patterns that the learning model hasn’t adapted to yet or even white / blacklists that aide the decision engine in handling high value customers. The working memory of the agent also contains instruction sets for arbitration to ensure that conflicting decisions are biased toward events of interest.

FEATURE EXTRACTION

As the agent is learning and adapting, it will generate new features from events as they are fed through the system. Think of the agent’s feature extraction as “automated feature engineering”, as time goes on new features are generated, older features are aged off, the result is a changing feature space that the learning model adapts to as it learns behaviors of interest.

LEARNING MODEL

A number of machine learning methods that can be employed as the learning model; however, in practice “semi-supervised” models tend to train and adapt best. Think of a semi-supervised model as a supervised model with little labeled data. SAS has developed both a unique supervised and semi-supervised models to combat fraud which have been tuned to identify rare events.

DECISION ENGINE & ARBITRATION

The outcome from the learning model is a score and an instruction set. The instruction set might be simple rule like IF score GT 600 THEN ALERT; or it can be a complex set of rules. Also within the decision engine user authored rules are executed. Sometimes user written rules conflict with what the

agent's learning model says to do. In those cases results are sent to arbitration. This is where the agent can be biased toward what the learning model says or what user authored rules say.

For example. A high value customer may in-advertently be flagged as an event of interest by the learning model and decision engine. User defined rules may flag this as a high value account in which case we don't' want to in-advertently shut their account down. Arbitration rules allow for biasing of results so that in some cases you may not want to raise an event of interest and in other cases you do.

ADAPTIVE LEARNING PROCESS

The learning process is not too different from a traditional machine learning process, it is just completely automated.

Figure 2. End to End Process

As figure 2 shows, the learning process is not really different from a manual model training process, it's just automated the phases. Here is an overview of how it works.

TRAINING PHASE

Given a training data set, with or without labeled data, the intelligent agent learns patterns within the data producing a "model of its environment:"

EVALUATION PHASE

The user evaluates the expected performance of the learning model of the environment, determines the critical factors and tailors the "instruction sets" & "interactions" to the agent's objective.

DEPLOYMENT PHASE

The model(s) and rules, when deployed, generate a decision pipeline that contains input vector (working memory), the learning model and a default event generating rules based on the instructions from the

learning phase. The agent takes in new events, scores them, and decides the events for review. New events of interest are fed into the learning model.

LEARNING PHASE

The agent combines “events of interest” and “normal” events from the system and folds it into working memory producing “training data”. The learning agent takes the data and learns to identify patterns of events of interest. As the agent learns from the events, it develops an instruction set to identify new and emerging events of interest. By combining supervised and unsupervised models in this way, the agent is able to adapt to new and unseen patterns within the data.

MONITORING

Autonomous agents require ongoing system and activity performance monitoring. Generally agents are self-sufficient, biasing toward supervised and unsupervised detection based on the data and activities within the system. System performance measures like how long it takes to make a decision (decision latency), how much resources it is using during the learning phase (resource consumption) and how much data it consumes, all need to be monitored over time. Activity performance is also key. It is important to track both the event-rate and the productivity rates over time to make sure the agent is actually doing something valuable. Finally these agents aren’t a panacea, overtime events of interest can change dramatically and the agent may need to be retired either for poor performance or be superseded by a better agent that is trained on different data.

SUPERVISED LEARNING

There are a number of supervised machine learning methods that can be used, however one of the more popular and accurate methods is to train an ensemble of multiple types of models, such as gradient boosted trees, random forests, support vector machines, and logistic regressions, with k-fold cross validation. The K-fold cross validation process finds the average estimated validation error (misclassification error for nominal targets or average square error for interval targets) for the trained model. During cross validation, all data are divided into k subsets (folds). For each fold, a new model is trained, and then validated using the selected fold. The validation error estimates are then averaged over each set of training and scoring executions to obtain a single value. In the next section, let us take a look at training gradient boosting trees model as an example.

Figure 3. K-Fold cross validation

TRAIN GRADIENT BOOSTED TREES WITH K-FOLD CROSS VALIDATION

Training a K-fold cross validated model is pretty simple. For this example, we are generating ten folds. A column named _fold_ is produced, to get repeatable folds copy the new table and add the _fold_ column. Here is an example of the CASL partition syntax which produces a dataset with a new column called _fold_:

```
partition /
    table = {
        name='training',
        compvars={'_fold_'},
        comppgm='call streaminit(__rankid*1000);
              _  fold_=floor(rand("UNIFORM")*10);'
      }
    outtable={name='training_w_fold', replace=True};
```

Train the gradient boosted decision tree by splitting the sub-sampled data, then splitting each resulting segment, and so on recursively until some validation constraint is met. Here a function is authored to train the gradient boosting model:

```
function OneFoldTree(nFold, iFold);
    /* generate _fold_ where */
    foldwhere1 = "_fold_ NE " || (String)iFold;
    foldwhere2 = "_fold_ EQ " || (String)iFold;
    mymodel = "gbt_" || (String)iFold;
    decisiontree.gbtreetrain  /
```

```
            table={name="training_w_fold", where=foldwhere1}
            inputs=xx
            target="FRAUD_FLAG"
            casout={name=mymodel, replace=1}
            maxbranch=2
            maxlevel=8
            leafsize=60
            ntree=100
            binorder=1
            nbins=100
            seed=1234
            learningRate=0.1
            subsamplerate=0.7
            m=64
        ;
        decisionTree.gbtreescore result = r/
            table = {name='training_w_fold', where=foldwhere2}
            model = {name=mymodel};
        print r;
        myscoredata = "gbtscore_" || (String)iFold;
        saveresult r  replace dataset=myscoredata;
    end func;
```

Next create a function to run each model in its own session:

```
    function KFoldCV(nFold);
        do i=1 to nFold;
            myerror[i] = OneFoldTree(nFold, i);
            end;
        return (myerror);
    end func;
```

During cross validation, all data are divided into k subsets (folds). For each fold, a new model is trained then validated using a selected fold. In this example, we have ten folds that we are going to train against the selected fold (nFold=1):

```
    nFold = 10;
    ModelError = KFoldCV(nFold);
    run;
```

Finally, compute the average error rate or the misclassification error or average square of k-fold cross validation like this:

```
mse  = 0;
do i= 1 to nFold;
  mse = mse + ModelError[i];
end;
run;
```

UNSUPERVISED LEARNING

There are a number of unsupervised methods for detecting anomalies, however one of the more interesting methods is a One-Class SVM. SAS provides a new procedure called SVDD (Support Vector Data Description) is a one-class classification technique that is useful in applications where data that belong to one class are abundant but data about any other class are scarce or missing. Fraud detection, equipment health monitoring, and process control are some examples of application areas where the majority of the data belong to one class. You can use SVDD to model such one-class data and subsequently use the model to perform outlier detection.

SVDD

The SVDD is currently available in SAS Visual Data Mining and Machine Learning 8.1 on SAS Viya as a CAS action set and a procedure SVDD procedure, including a prebuilt drag-and-drop SVDD task. All three approaches are really easy to use. The procedure is pretty handy at uncovering anomalies example syntax:

```
PROC SVDD data=casuser.training_w_fold OUTLIER_FRACTION=number;
    input &numeric_vars / level=interval
          &cat_vars / level=nominal;
    id account_no;
    solver actset;
    kernel rbf / bw=&optimal;
    savestate rstore=casuser.training_state;
run;
```

There are a couple parameters that are key to the procedure. Frist the outlier_fraction, by default this is 1E–6, you can adjust this to be more in line with expected volume of identified outliers. Second is the Kernel settings. Currently only Radial Basis Function (RBF) is supported. You need to set a bandwidth parameter “s” where which must be a positive nonzero number.

$$K(x,y) = \exp \frac{-\|x-y\|^2}{2s^2}$$

The bandwidth parameter “s” impacts the kernel-parameter, as “s” is decreased the processing time increases.

WHITEBOX-ING

Machine learning systems generally fail to focus on the consumers of their models, instead focusing on the minimization of error and establishing bounds on their error (Vapnik 1996). The use of surrogate models is becoming a more popular practice in machine learning. Surrogate models enable the modeler to build an explainable “white-box” model to approximate what the “black-box” machine learning model is doing behind the scenes. Decision trees and logistic regression are often used to approximate complex machine learning methods, but lack the interpretably for a fraud examiner, to address this gap the SAS intelligent agent generates a visual scorecard and narrative explaining the interactions for each “Scored” observation.

SIMPLE SCORECARD

By default the intelligent agent’s decision engine creates a visual scorecard explaining the approximation of the score that the observation received. Figure 4 is a representative example of what an investigation analyst might view as the result of an event of interest.

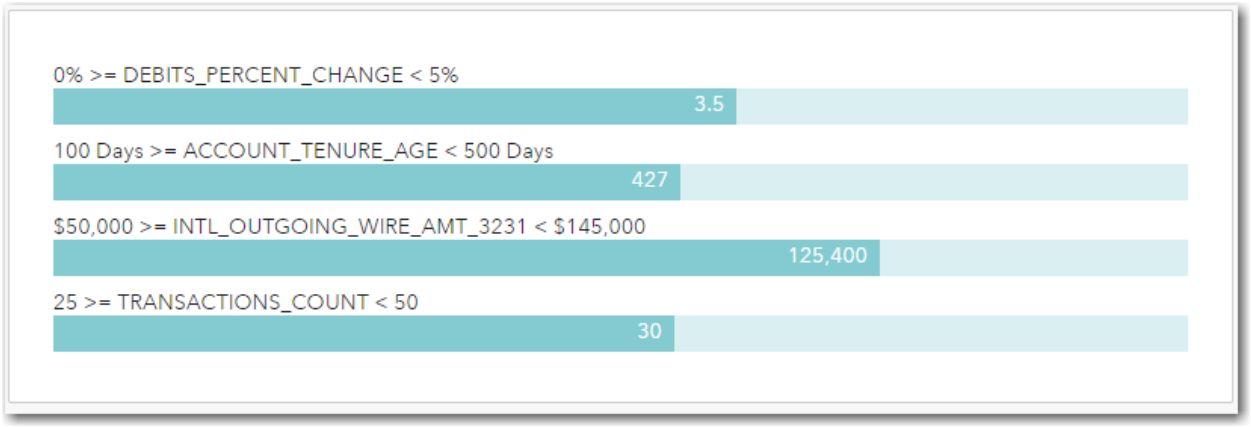

Figure 4. scorecard

NARRATIVE

The intelligent agent also can generate human readable “narratives” of the activity of interest. This is useful for both documentation and for accessibility since the text can also be turned to voice relatively easily. Figure 5. Provides a sample narrative of an event of interest.

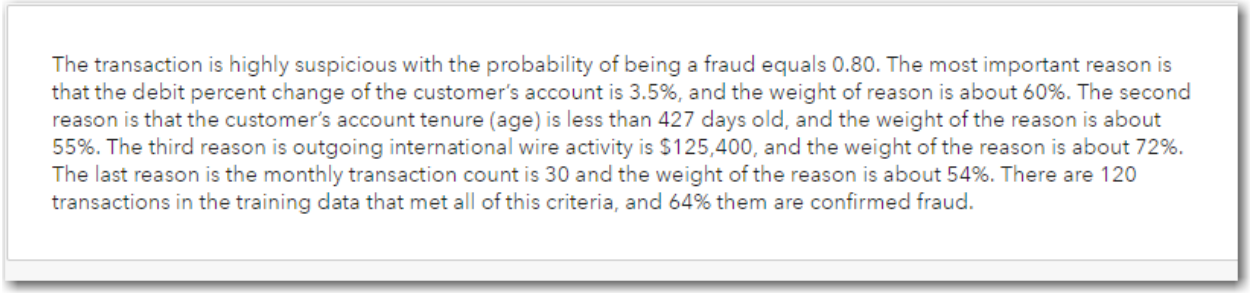

Figure 5. narrative

CONCLUSION

As automation and the use of intelligent agents rise, industry will need to deal with the results of the impact of automation on their business processes. While the advent of machine learning “bots” for combating fraud, waste, and abuse of data is new, the use of machine learning to combat fraud goes back decades there are two keys to deploying machine learning agents to combat fraud, waste and abuse. First you need continuous monitoring and evaluation of the learning process, these agents can’t

be left to learn on their own, rather it will require some qualitative assessment of the impact of these agents. Second, the consumption of machine learning outputs for fraud, waste and abuse has to be considered. As the level of sophistication of machine learning increases the ability to consume and assess what the "machine" is doing is key to the success.

REFERENCES

Diana Shaw, "Condition Monitoring and Anomaly Detection for High-Frequency Multivariate IoT Data", May 2017, available at https://communities.sas.com/t5/SAS-Communities-Library/Condition-Monitoring-and-Anomaly-Detection-for-High-Frequency/ta-p/371783

Sun, R., and Tsung, F. (2003). "A Kernel-Distance-Based Multivariate Control Chart Using Support Vector Methods." *International Journal of Production Research* 41:2975–2989.

Vapnik, V.M. (1996). The Nature of Statistical Learning Theory. New York: Springer-Verlag.

ACKNOWLEDGMENTS

I would like to thank and acknowledge Dr. Yue Qi, Jeff Miller, Tom Mutdoch, Iain Jackson, Rory MacKensie and Gordon Robinson for their support and efforts to develop the SAS Adaptive Learning and Intelligent Agent System.

RECOMMENDED READING

- SAS Visual Data Mining and Machine Learning 8.2: Procedures

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Michael Ames
SAS Institutue
michael.ames@sas.com