×

You currently do not have any widgets inside Off-Canvas sidebar. Add widgets in Appearance > Widgets > Off-Canvas Sidebar.

×

Search

×

- Platform

EDGEPLATFORM

The RiskSpan Edge Platform is a module-based data management, modeling, and predictive analytics software platform for loans and fixed-income securities. Our scalable, cloud-based platform enables you to make better business decisions based on uncommon insights into historical trends and advanced predictive forecasts.

Edge Platform Modules:

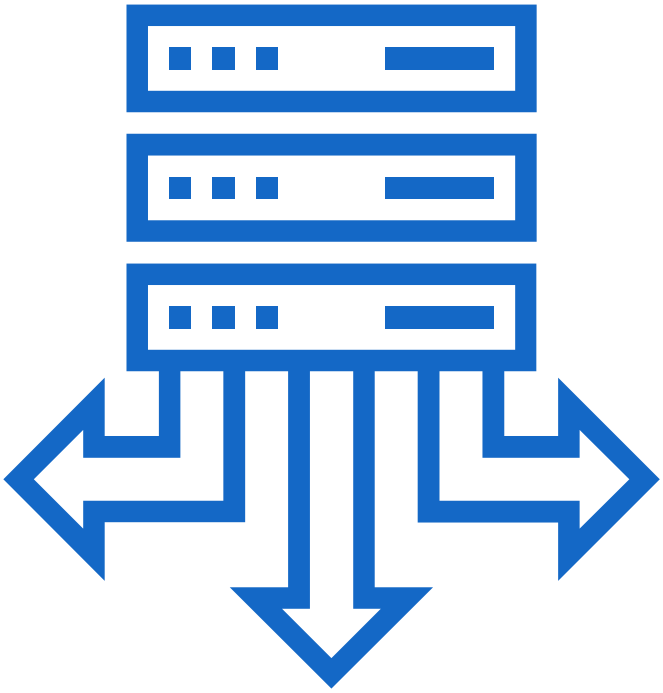
- Agency-MBS Trader Module
- Structured Products Module
- Whole Loan Module
- Forecasting (CECL) Module
- Market Risk Service Module
- Model Risk Management Module

- Services

PREDICTIVE ANALYTICS

Make informed data-driven decisions with powerful forecasts and predictive analytics.

Learn More



DATA MANAGEMENT

Knock down silos – enable access, movement, and blending of source data for downstream analytics.

Learn More

MODEL AND DATA GOVERNANCE

Get the most out of your model risk management and data governance programs.

Learn More

Consulting Services Expertise:

- Predictive Analytics
- Data Management
- Model and Data Governance
- Machine Learning

October 25, 2017by Joshua Falter in Data & Analytics

Evaluating Supervised and Unsupervised Learning Models

Model evaluation (including evaluating supervised and unsupervised learning models) is the process of objectively measuring how well machine learning models perform the specific tasks they were designed to do—such as predicting a stock price or appropriately flagging credit card transactions as fraud. Because each machine learning model is unique, optimal methods of evaluation vary depending on whether the model in question is "supervised" or "unsupervised." Supervised machine learning models make specific predictions or classifications based on labeled training data, while unsupervised machine learning models seek to cluster or otherwise find patterns in unlabeled data.

Unsupervised Learning

Common unsupervised learning techniques include *clustering*, *anomaly detection*, and *neural networks*. Each technique calls for a different method of evaluating performance. We'll focus on clustering models as an example. Clustering is the task of grouping a set of objects in such a way that objects in the same cluster are more like each other than they are to objects in other clusters. Various algorithms are capable of clustering, including *k-means* and *hierarchical*, which differ in their definitions of a cluster and how to find one.

Evaluating Unsupervised Learning Models

Let's assume that we need to cluster banking customers together into groups based on the amount and magnitude of risk they pose. After the clustering algorithm has grouped the customers into distinct clusters, we need to evaluate how well those clusters were formed. The lack of labels on an unsupervised learning model's training data makes evaluation problematic because there is nothing to which the model's results can be meaningfully compared. If we were to manually group these customers, we could then compare our manual groupings with the algorithm's, but often this is not an option due to time or labor constraints, so we need a more efficient way to determine how well the algorithm performed.

One way would be to determine 1) how close each customer within each cluster is to every other customer in its cluster (the "intra-cluster" distance") and 2) how close each cluster of customers is to other clusters (the "inter-cluster" distance), and then to compare the two distances. Models that produce relatively small intra-cluster distances and relatively large inter-cluster distances evaluate favorably because they appear to be doing a good job of grouping like customers with discrete characteristics.

Supervised Learning

Within supervised learning, there are techniques for both regression and classification tasks. While some techniques are suited to either regression or classification, some can be used for both. For example, *linear regression* can only be used for regression while *support vector machines* and *random forests* can be used for either. While each of these is a different technique, the metrics that we use to evaluate them are the same, so we can even compare these models to one another. In our examples, we'll focus on flagging credit card purchases as fraud, a classification task, and predicting housing prices, a regression task.

Evaluating Supervised Models

The task of evaluating how well a supervised learning model performs is more straightforward. Because supervised learning models learn from labeled training data, once they have been fitted using training data, they can be tested against data from the same population and therefore has the same labels.

For example, let's say we need to classify whether a credit card transaction is fraudulent and we have a dataset of transactions with labels of either "fraud" or "not fraud." We can (and sometimes do[1]) train our model on all the available data, but this prevents us from fairly evaluating it because no "independent" data remains for testing and overfitting[2] becomes difficult to detect. This problem can be avoided by splitting the available data into training and testing sets.

This can be accomplished in various ways. For simplicity, we'll first talk about splitting our dataset into two sets: a training set (typically 70% of the whole dataset) from which the model learns and a test set (the other 30%). Because the test set is withheld from the model during training, it can contribute to an unbiased evaluation of how well a model performs on previously unseen data. This protects against overfitting and allows us to evaluate how our model would perform "in the wild" on new data as it emerges.

Cross-validation is another antidote for overfitting. Cross-validation involves partitioning data into multiple groups and then training and testing models on different group combinations. For example, in a 5-fold cross-validation we would split our transaction data set into five partitions of equal sizes. We would then train our model on four of those five partitions and test our model on the remaining partition. We would then repeat the process—selecting a different partition to be the test group and training a new model on the remaining set of four partitions. We would repeat three more times, for a total of five rounds of cross-validation, one for each fold. We will then have five different models, each having been trained and tested on a different subset of data and each having their own weights and prediction accuracy. At the end, we combine these models by averaging their weights together to estimate a final predictive model.

Classification metrics are the measures against which models are evaluated. The simplest and most common such metric is accuracy. Accuracy is computed by dividing the number of correct predictions by the total number of predictions. In our supervised transaction classification model example, if we tested our model on one hundred transactions and correctly predicted their label (fraud/not fraud) for ninety-five of them, then the accuracy of our model is 95%.

Accuracy is the simplest, most understandable metric we can use, but we wouldn't want to rely on accuracy alone because it doesn't distinguish between false positives, transactions incorrectly classified as fraud, and false negatives, transactions incorrectly classified as non-fraud. For this we need a confusion matrix.

A confusion matrix is a 2-by-2 table that sorts predictions into one of four classifications: true positive, true negative, false positive, and false negative. Our transaction classification model might generate a confusion matrix like this one:

		Actual	
		Fraud	Not Fraud
Predicted	Fraud	4 (True Positive)	2 (False Positive)
	Not Fraud	3 (False Negative)	91 (True Negative)

The confusion matrix indicates that, out of 100 total transactions, our model correctly predicted fraud four times and correctly predicted not fraud 91 times, yielding an overall accuracy of 95%. The confusion matrix, however, also enables us to see the number of times the model incorrectly predicted that a transaction was fraud—a false positive which occurred on two out of the 100 transactions. We can also see the number of times the model predicted a transaction was not fraud when it was—a false negative which occurred on three out of the 100 transactions.

While the model appears to boast a fairly strong "true negative" rate—the percentage of non-fraud messages correctly classified as such (91/(91+2)=97.8%), the model's "true positive" rate—the percentage of fraud messages correctly flagged as such (4/(4+3)=57.1%) is far less attractive. Breaking down the model's performance in this way paints a different and more complete picture than the 95% accuracy rate alone.

Evaluation methods apply to regression models, as well. Let's assume we have a regression model that's been trained to predict housing prices. The model's predicted prices can be compared with actual prices using the mean squared-error, which measures the average of the squares of the errors, which are the differences between the actual and predicted price. The lower the mean squared-error, the better the model.

All models need to be subjected to evaluation—when they are built and throughout their lives. Supervised and unsupervised learning models pose different sorts of evaluation challenges, and selecting the right type of metrics is key.

---

[1] Many fraud detection models are also built using neural networks and other *unsupervised* learning techniques.

[2] Overfitting occurs when a model makes decisions with powerful predictive analytics based on data points that are not germane to the analysis. Continuing the example of fraud detection, overfitting may occur if model training detects a correlation between the length of a customer's name (or whether the customer's name begins with a vowel) and the likelihood that a transaction is fraudulent. Testing is likely to expose random, spurious correlations of this type for what they are, as they are not likely to be replicated in the test data set that has been held out from the training data. A model that has been "overfit" to its training data is likely to return a considerably lower accuracy ratio on the test data.

---

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

http://www.oreilly.com/data/free/files/evaluating-machine-learning-models.pdf

https://en.wikipedia.org/wiki/Cluster_analysis#Evaluation_and_assessment

http://www.mit.edu/~9.54/fall14/slides/Class13.pdf

https://stats.stackexchange.com/questions/79028/performance-metrics-to-evaluate-unsupervised-learning

machine learning
Share on Twitter
Share on Facebook
Share on Google+
Share on Pinterest

Search

Categories

- Conferences & Events (7)
- Data & Analytics (32)
- Governance & Compliance (33)
- Market Risk (4)
- Media (16)
- Mortgage Market (29)

Tags

Agile AML back-testing benchmarking big data blockchain capital planning CECL credit risk modeling credit risk transfer CSS data management DevOps DFAST / CCAR EUC FASB FHA FHFA FHLBanks GSEs HARP IFRS 9 interest rate models interview machine learning MBS modeling model risk management model validation mortgage insurance non-agency MBS NPLs open source portfolio analytics press release private-label securities rs edge securitization spreadsheets strest testing student loans VA loans VaR vega

Platform

- Agency-MBS Trader Module
- Structured Products Module
- Whole Loan Module
- Forecasting (CECL) Module
- Market Risk Service Module
- Model Risk Management Module

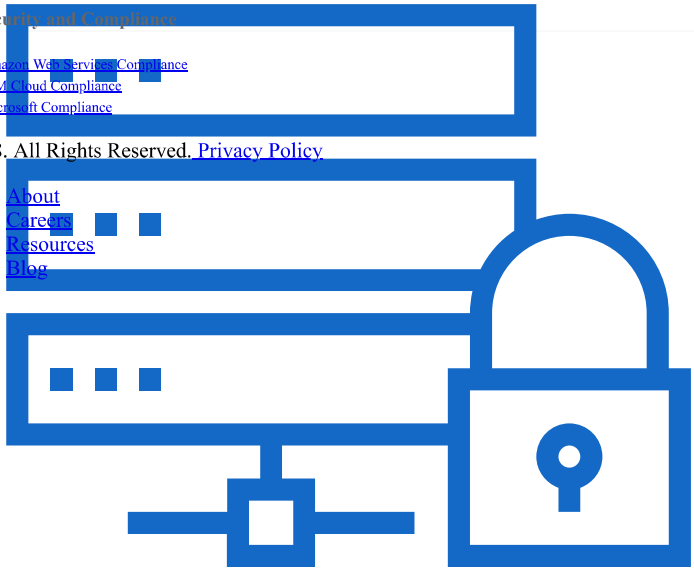
Services

- Predictive Analytics
- Data Management

- Model and Data Governance
- Machine Learning

Information Security and Compliance

- Amazon Web Services Compliance
- IBM Cloud Compliance
- Microsoft Compliance



MODEL AND DATA GOVERNANCE

Get the most out of your model risk management and data governance programs.

Learn More

Consulting Services Expertise:

- Predictive Analytics
- Data Management
- Model and Data Governance
- Machine Learning

- About
  - Story
  - Team
  - Join Us
  - Contact
- Resources
- Blog