Applications of Machine Learning to Risk Management

MF 731 Corporate Risk Management

Introduction/Outline

Machine Learning (ML) is widely used in risk management.

Lecture Goals

- Learn the basics of supervised ML for classification.
  - k Nearest Neighbors; Logistic Regression; Linear Discriminant Analysis; Support Vector Machines.
  - (sadly) No time for neural networks; decision trees, etc..
- Consider two applications to risk management.
  - Credit Scoring: identifying credit-worthy applicants.
  - Fraud Detection: identifying fraudulent transactions.

The Classification Problem

Goal: predict, based upon *data features*, to which class the data belongs.

- Credit scoring: use applicant data such as income and marital status to predict credit-worthiness.
- Fraud detection: use financial transactions data to identify money laundering.

We want the machine to learn!

- Use *training* data to accurately classify new data.
    - This is *supervised* learning.

We will focus on binary classification problems, which are natural for risk management.

Notation

X: feature data.

Credit scoring: annual income, marital status, number of dependents, etc.

We assume M features, so that $X \in \mathbb{R}^M$.

Data may be numeric or categorical.

Examples (categorical): marital status, occupation.

Examples (numeric): annual income, monthly credit card debt.

G: classification data.

Two classes. $G = 1$ or $G = 0$.

Training Data and Learning

We use *training data* to train the machine.

$\{(x_n, g_n)\}_{n=1}^N$. x_n: feature, g_n: class.

Plan:

Use $\{(x_n, g_n)\}$ to build a map $G : \mathbb{R}^M \to \{1, 0\}$.

Classify a new data point by $g = G(x)$.

There are many methods to do this. We will give overviews for

k nearest neighbors, logistic regression, linear discriminant analysis, support vector machines.

k Nearest Neighbors (kNN)

Idea: for $x \in \mathbb{R}^M$ write $(x_{n_1}, \dots, x_{n_k})$ as the k nearest points in the training set.

"Nearest": typically use Euclidean distance.

Classify x by a "majority vote" on the class of $\{x_{n_j}\}$, with ties broken arbitrarily.

$k = 5$: classify to $0, 1$ based upon which class has ≥ 3 training data points.

$k = 10$: classify based on which class has ≥ 6 training data points. If each has 5, pick a class at random.

kNN: Notes

Training data is usually adjusted so that (each component) has zero sample mean and unit variance.

Because data may be measured in different units.

One must be careful assigning numeric values to categorical data.

Many techniques developed to address this issue.

Adjustments must be made in high dimensions.

Nearest neighbors are typically quite far away.

Notion of "distance" must be adjusted.

Method works well with irregular decision boundaries (e.g. classifying handwritten numbers).

Logistic Regression

Logistic regression specifies the *posterior* prob.

$$\mathbb{P}\left[G = 1 \mid X = x\right] = \frac{e^{\tilde{x}'\beta}}{1+e^{\tilde{x}'\beta}} =: p(\tilde{x};\beta).$$

$\tilde{x} = \left(\begin{smallmatrix} 1 \\ x \end{smallmatrix}\right)$, $\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$ so that $\tilde{x}'\beta = \beta_0 + x'\beta_1$

$a'b = \sum_{m=1}^{M} a^{(m)} b^{(m)}$ is the dot product.

We use MLE to identify the optimal $\widehat{\beta}$.

Classification rule: $\widehat{G}(x) = 1_{p(x;\widehat{\beta}) \geq .5} = 1_{\tilde{x}'\widehat{\beta} \geq 0}$.

Linear decision boundary: $\left\{\tilde{x}'\widehat{\beta} = 0\right\}$.

"Logistic": affine *logit* transformation of the posterior probabilities

logit transform = $\frac{x}{1-x}$

$$\log\left(\frac{\mathbb{P}\left[G=1 \mid X=x\right]}{1-\mathbb{P}\left[G=1 \mid X=x\right]}\right) = \log\left(\frac{p(x;\beta)}{1-p(x;\beta)}\right) = \tilde{x}'\beta$$

$$\ell(B) = \log\left(\prod_{n=1}^{N} \mathbb{P}(G = g_n \mid X = x_n)\right)$$

$$= \sum_{n=1}^{N} g_n \log(p(x_n, \beta)) + (1-g_n)\log(1 - p(x_n, \beta))$$

$$= \sum_{n=1}^{N} g_n\left(\hat{x}_n'\beta - \log(1+e^{\hat{x}_n'\beta})\right) + (1-g_n)\left(0 - \log(1+e^{\hat{x}_n'\beta})\right)$$

$$= \sum_{n=1}^{N} g_n \hat{x}_n'\beta - \log(1+e^{\hat{x}_n'\beta})$$

Logistic Regression

We find $\hat{\beta}$ using MLE on the posterior probabilities given the training data $\{(x_n, g_n)\}$.

$$\ell(\beta) = \log\left(\prod_{n=1}^N \mathbb{P}\left[G = g_n \mid X = x_n\right]\right)$$

$$= \sum_{n=1}^N \left(g_n \log(p(x_n;\beta)) + (1-g_n)\log(1-p(x_n;\beta))\right).$$

$$= \sum_{n=1}^N \left(g_n\, \widetilde{x}_n'\beta - \log\left(1 + e^{\widetilde{x}\cdot_n\beta}\right)\right).$$

By concavity, $\widehat{\beta}$ solves the non-linear equations

$0 = \sum_{n=1}^{\tilde{N}} g_n \tilde{x}_n - \tilde{x}_n P(x_n, \beta)$

$$0 = \nabla_\beta \ell(\beta) = \sum_{n=1}^N \left(g_n - p(x_n;\beta)\right)\widetilde{x}_n.$$

$\widehat{\beta}$ practically found using built-in commands.

Linear Discriminant Analysis (LDA)

LDA starts with Bayes' rule, which implies

$$\mathbb{P}\left[G=1 \mid X=x\right] = \frac{\pi f_1(x)}{\pi f_1(x)+(1-\pi)f_0(x)}$$

$\mathbb{P}(G=1|X\in A) = \frac{\mathbb{P}(G=1, X\in A)}{\mathbb{P}(X\in A)}$

$\pi = \mathbb{P}[G=1]$, and f_i is the pdf of $X|G=i$.

$= \frac{\mathbb{P}(X\in A|G=1)\,\mathbb{P}(G=1)}{\mathbb{P}(X\in A|G=1)\mathbb{P}(G=1)+\mathbb{P}(X\in A|G=0)\mathbb{P}(G=0)}$

LDA then assumes $X|G=i \sim N(\mu_i, P^{-1})$

$P=\Sigma^{-1}$: precision matrix. f_i: $N(\mu_i, P^{-1})$ pdf.

Logit transform: $\log\left(\frac{\mathbb{P}[G=1|X=x]}{1-\mathbb{P}[G=1|X=x]}\right) = \widetilde{x}'\alpha = \binom{1}{x}'\binom{\alpha_0}{\alpha}$

$$\alpha_0 = \log\left(\frac{\pi}{1-\pi}\right) - \tfrac{1}{2}\mu_1' P\mu_1 + \tfrac{1}{2}\mu_0' P\mu_0, \quad \alpha_1 = P(\mu_1-\mu_0).$$

Classification is similar to logistic regression.

$\widehat{G}(x) = 1_{\mathbb{P}[G=1|X=x]\geq .5} = 1_{\widetilde{x}'\alpha\geq 0}$ (linear boundary).

$$\mathbb{P}(G=1 \mid X=x) = \frac{\pi f_1(x)}{\pi f_1(x) + (1-\pi) f_0(x)}$$

$$= \frac{\pi f_1(\mu_1) e^{-\frac{1}{2}x'\Sigma^{-1}x + x'\Sigma^{-1}\mu_1}}{\pi f_1(\mu_1) e^{-\frac{1}{2}x'\Sigma^{-1}x + x'\Sigma^{-1}\mu_1} + (1-\pi) f_0(\mu_0) e^{-\frac{1}{2}x'\Sigma^{-1}x + x'\Sigma^{-1}\mu_0}}$$

$$= \frac{\pi f_1(\mu_1) e^{x'\Sigma^{-1}\mu_1}}{\pi f_1(\mu_1) e^{x'\Sigma^{-1}\mu_1} + (1-\pi) f_0(\mu_0) e^{x'\Sigma^{-1}\mu_0}}$$

$$\operatorname{Log}(\mathbb{P}(G=1 \mid X=x)) = \overbrace{\operatorname{Log}\left(\frac{\pi}{1-\pi} \frac{f_1(\mu_1)}{f_0(\mu_0)}\right)}^{\alpha_0} + \underbrace{x'\Sigma^{-1}(\mu_1-\mu_0)}_{\alpha_1}$$

$$= \alpha_0 + x'\alpha_1$$

$$= \tilde{x}'\alpha \qquad \text{where } \tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix} \quad \alpha = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

Estimating the parameters

Given the training data, (π, μ_1, μ_0, P) are found using MLE on the *unconditional* joint density

$$\tfrac{d}{dx}\mathbb{P}\left[X \in dx, G = k\right] = \pi f_1(x) + (1-\pi) f_0(x).$$

This yields the explicit answers

$$\widehat{\pi} = \tfrac{1}{N}\sum_{n=1}^{N} g_n,\ \widehat{\mu}_1 = \frac{\sum_{n=1}^{N} g_n x_n}{\sum_{n=1}^{N} g_n},\ \widehat{\mu}_0 = \frac{\sum_{n=1}^{N}(1-g_n)x_n}{\sum_{n=1}^{N} 1-g_n}.$$

$$\widehat{P}^{-1} = \tfrac{1}{N}\sum_{n=1}^{N}\left(g_n(x_n-\widehat{\mu}_1)(x_n-\widehat{\mu}_1)' + (1-g_n)(x_n-\widehat{\mu}_0)(x_n-\widehat{\mu}_0)'\right)$$

LDA versus Logistic Regression

LDA and Logistic Regression appear very similar.

Logistic: $\log\left(\frac{\mathbb{P}[G=1|X=x]}{1-\mathbb{P}[G=1|X=x]}\right) = \widetilde{x}'\widehat{\beta},\ \widehat{G}(x) = 1_{\widetilde{x}'\widehat{\beta}\geq 0}$.

LDA: $\log\left(\frac{\mathbb{P}[G=1|X=x]}{1-\mathbb{P}[G=1|X=x]}\right) = \widetilde{x}'\widehat{\alpha},\ \widehat{G}(x) = 1_{\widetilde{x}'\widehat{\alpha}\geq 0}$.

The difference is in how $\widehat{\beta}, \widehat{\alpha}$ are obtained.

Logistic: MLE on the cond. prob. $\mathbb{P}[G = k|X = x]$.

LDA: MLE on the uncond. prob. $\mathbb{P}[G = k, X \in dx]$ under a Gaussian assumption for X given G.

Notes on LDA

There is a rich theory for LDA. Selected highlights

"Discriminant": $1_{\tilde{x}'\alpha>0} \Leftrightarrow 1_{\delta_1(x)\geq\delta_0(x)}$ where

$$\delta_i(x) = x'P\mu_i - \tfrac{1}{2}\mu_i'P\mu_i + \log(\pi_i)\ (\pi_1 = \pi, \pi_0 = 1-\pi).$$

Linear "discriminant" function.

Quadratic disc. analysis (QDA): $X|G = i \sim N(\mu_i, P_i^{-1})$.

I.e. we allow the precision matrix to vary.

$$\widehat{G}(x) = 1_{\mathbb{P}[G=1|X=x]\geq .5} = 1_{\delta_1(x)\geq\delta_2(x)}$$

$$\delta_i(x) = \log(\pi_i) + \tfrac{1}{2}\log(|P_i|) - \tfrac{1}{2}(x-\mu_i)'P_i(x-\mu_i).$$

Quadratic decision boundary $\{\delta_1(x) = \delta_2(x)\}$.

Support Vector Machines (SVM)

Each decision boundary for the above methods take the form $\{\beta_0 + x'\beta_1 = 0\}$.

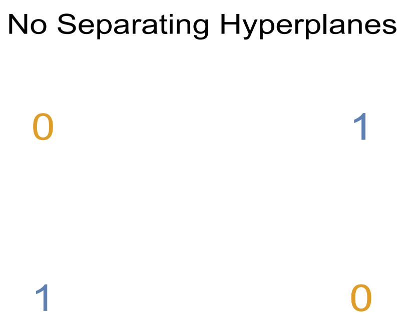
The set $\{\beta_0 + x'\beta_1 = 0\}$ is a M dimensional *hyperplane*.

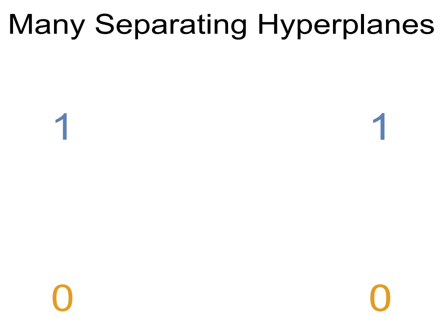
SVM starts with the idea of classifying points based "upon what side" of a hyperplane they lie.

Provided the data can be separated.

If the data cannot be separated, adjustments are made.

Support Vector Machines (SVM)





If separable, typically $\exists$ *many* separating hyperplanes.

Is there a “best” one? If so, how can we find it?

SVM: Separable Case

Assume the data can be separated.

There is some β_0, β_1 such that for $n = 1, ..., N$

$$g_n = 1 \Rightarrow \beta_0 + x_n'\beta_1 > 0 \text{ and } g_n = 0 \Rightarrow \beta_0 + x_n'\beta_1 < 0.$$

Notation adjustment: change $g_n = 0$ to $g_n = -1$.

Then $g_n(\beta_0 + x_n'\beta_1) > 0, n = 1, ..., N$ so that $\exists\ M > 0$ s.t. $g_n(\beta_0 + x_n; \beta_1) \geq M$ for all n.

"Best": find $\widehat{\beta}_0, \widehat{\beta}_1$ $(|\widehat{\beta}_1| = 1)$ with the largest M.

Problem has no answer if we do not restrict size of β_1.

SVM: Separable Case

Optimization problem

$$\max_{\beta_0,\beta_1,|\beta_1|=1} M \text{ s.t. } g_n(\beta_0 + x_n'\beta_1) \geq M.$$

Note: inequalities assumed to hold for $n = 1, ..., N$.

Convenient transformation

$$g_n(\beta_0 + x_n'\beta_1) \geq M \text{ implies } g_n(\widetilde{\beta}_0 + x_n'\widetilde{\beta}_1) \geq 1.$$

$\widetilde{\beta}_i = \beta_i / M$ for $i = 0, 1$, so $|\widetilde{\beta}_1| = 1/M$.

M large $\Leftrightarrow$ $|\widetilde{\beta}_1|$ small. $\longleftarrow$

This leads to the equivalent problem

$$\min_{\beta_0,\beta_1} \tfrac{1}{2}|\beta_1|^2 \text{ s.t. } g_n(\beta_0 + x_n'\beta_1) \geq 1.$$

$$\min_{\beta_0,\beta_1} \frac{1}{2}|\beta_1|^2 \text{ s.t. } g_n(\beta_0 + x_n'\beta_1) \geq 1$$

To solve this problem we use Lagrange multipliers

$$\min_{\beta_0,\beta_1,\alpha} \frac{1}{2}|\beta_1|^2 - \sum_{n=1}^{N} \alpha_n \left(g_n(\beta_0 + x_n'\beta_1) - 1\right) \text{ s.t.}$$

$$\alpha_n \geq 0;\ g_n(\beta_0 + x_n'\beta_1) \geq 1;\ \alpha_n \left(g_n(\beta_0 + x_n'\beta_1) - 1\right) = 0.$$

First order optimality conditions

$$(\nabla_{\beta_1} = 0) \qquad \widehat{\beta}_1 = \sum_{n=1}^{N} \widehat{\alpha}_n g_n x_n.$$

$$(\nabla_{\beta_0} = 0) \qquad 0 = \sum_{n=1}^{N} \widehat{\alpha}_n g_n.$$

$$(\nabla_{\alpha} = 0) \qquad \widehat{\alpha}_n > 0 \Leftrightarrow g_n(\widehat{\beta}_0 + x_n'\widehat{\beta}_1) = 1.$$

$$\widehat{\beta}_1 = \sum_{n=1}^{N} \widehat{\alpha}_n g_n x_n, \qquad \widehat{\alpha}_n > 0 \Leftrightarrow g_n(\widehat{\beta}_0 + x_n'\widehat{\beta}_1) = 1$$

We then classify a new point x by

$$\widehat{G}(x) = \operatorname{sign}\left(\widehat{\beta}_0 + x'\widehat{\beta}_1\right) = \operatorname{sign}\left(\widehat{\beta}_0 + \sum_{n=1}^{N} \widehat{\alpha}_n g_n x' x_n\right).$$

$$\operatorname{sign}(z) = 1_{z\geq 0} - 1_{z<0}.$$

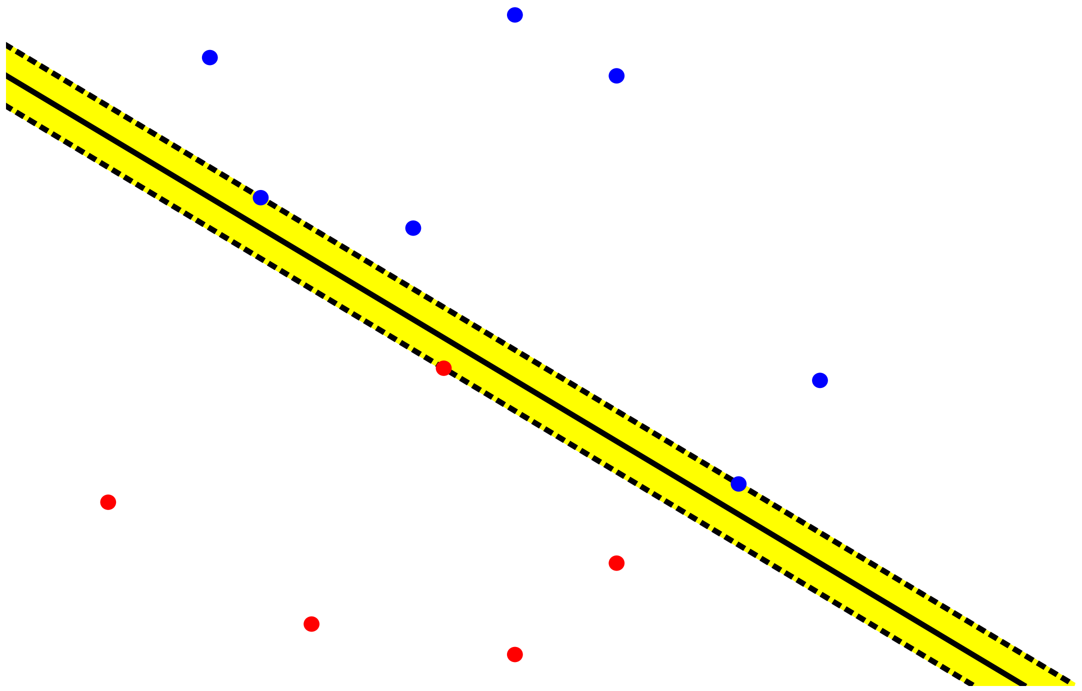
"Support vector machine"

Optimal $\hat{\beta}$ is a linear comb. of those x_n for which $\widehat{\alpha}_n > 0$.

But, $\alpha_n > 0$ only for (x_n, g_n) on the *slab* $g_n(\widehat{\beta}_0 + x_n'\widehat{\beta}_1) = 1$.

Optimal hyperplane "supported" by the vectors on the slab.

SVM in a Picture



SVM: Non-Separable Case

In reality the data will almost never be separable.

I.e. for any β_0, β_1, $\exists$ n s.t. $g_n(\beta_0 + x_n'\beta_1) < 0$.

Work around: introduce *slack* variables $\xi_n > 0$

$$\min_{\beta_0, \beta_1, \xi} \tfrac{1}{2}|\beta_1|^2 + C \sum_{n=1}^{N} \xi_n \quad \text{s.t.} \quad \xi_n \geq 0, g_n(\beta_0 + x_n'\beta_1) \geq 1 - \xi_n \ .$$

Slack: we now only require $g_n(\beta_0 + x_n'\beta_1) \geq 1 - \xi_n$.

C: cost parameter which controls penalization of large ξ_n.

Non-Separable Case: Solution

As before, $\widehat{\beta}_1 = \sum_{n=1}^{N} \widehat{\alpha}_n g_n x_n$. But now,

If $g_n\left(\widehat{\beta}_0 + x_n'\widehat{\beta}_1\right) > 1$ then $\widehat{\alpha}_n = 0$.

If $g_n\left(\widehat{\beta}_0 + x_n'\widehat{\beta}_1\right) = 1$ then $0 < \widehat{\alpha}_n < C$.

If $g_n\left(\widehat{\beta}_0 + x_n'\widehat{\beta}_1\right) < 1$ then $\widehat{\alpha}_n = C$.

$\widehat{\beta}_1$ supported by x_n s.t. $g_n\left(\widehat{\beta}_0 + x_n'\widehat{\beta}_1\right) \leq 1$

If no slack needed $(= 1)$ we give a smaller weight in $(0, C)$.

If slack needed (< 1) we give the maximal weight C.

Classification same as before

$$\widetilde{G}(x) = \operatorname{sign}\left(\widehat{\beta}_0 + x'\widehat{\beta}_1\right) = \operatorname{sign}\left(\widehat{\beta}_0 + \sum_{n=1}^{N} \widehat{\alpha}_n g_n x' x_n\right).$$

SVM and Kernels

Write $\langle x, x_n \rangle := x^{\mathrm{T}} x_n$ (inner product). Our rule is

$$\widetilde{G}(x) = \operatorname{sign}\left(\widehat{\beta}_0 + \sum_{n=1}^{N} \widehat{\alpha}_n g_n \langle x, x_n \rangle\right).$$

To improve performance, we may *transform* the data by a map $h : \mathbb{R}^M \to \mathbb{R}^M$ and apply SVM with h

I.e. we set $y_n := h(x_n)$ and classify using $\{(y_n, g_n)\}_{n=1}^{N}$.

Identify $\widehat{\beta}_0, \widehat{\beta}_1, \left\{\widehat{\xi}_n\right\}, \{\widehat{\alpha}_n\}, \{\widehat{\mu}_n\}$ using $\{(y_n, g_n)\}$.

Classify x by $\widetilde{G}(x) = \operatorname{sign}\left(\widehat{\beta}_0 + \sum_{n=1}^{N} \widehat{\alpha}_n g_n \langle h(x), h(x_n) \rangle\right)$.

SVM and Kernels

Define the *kernel* $K(x_1, x_2) := \langle h(x_1), h(x_2) \rangle$.

For classification, only K matters.

$$\widetilde{G}(x) = \operatorname{sign}\left(\widehat{\beta}_0 + \sum_{n=1}^{N} \widehat{\alpha}_n g_n K(x, x_n)\right).$$

In fact, only K matters for identifying $\widehat{\alpha}$ as well.

Popular choices for K

Euclidean: $K(x_1, x_2) = x_1' x_2$ (what we used).

Radial basis: $K(x_1, x_2) = e^{-\gamma |x_1 - x_2|^2}$.

Neural : $K(x_1, x_2) = \tanh(\kappa_1 x_1' x_2 + \kappa_2)$

Application: Credit Scoring

We now apply the above theory to credit scoring.

Question: should we loan someone money?

- Goal: use ML to classify as creditworthy (1) or not (0).

Historically:

- Loans were given by individuals at banks.
- At least in the US, systematic bias and racism led to denials of credit based for non-financial reasons.
- Idea: loans given via ML may be less discriminatory.
    - Some literature refutes this (not our focus today).

Feature Data

Two data sets with 13 and 20 features respectively.

- Each data set had around 1200 applicants.
- Issue: obtaining quality data in sufficient volume.

Example: 13 feature set

- Age; # of kids; # of dependents; Home phone?; Spouse income; Employment type; Annual income; Residential status; Home value; Mortgage balance; Mortgage/rent payments; Other loan payments; Credit card payments.

Categoric Examples

- Home phone (yes or no), Employment type (government, private sector, retired, student, etc.).
- Delicate issues giving categoric variables numeric values.

Results

We used the following classification methods

Logistic regression; LDA; SVM (radial kernel); kNN.

Conclusion

Methods gave comparable performance.

Classification success rates of around 70% – 80%.

Separated data into *training*, *tuning*, *testing* subsets.

All methods ran very quickly (skLearn in Python).

Operational Risk

ML classification is used in Operational Risk as well.

Goal: classify a financial transaction as fraudulent.

Abstractly: many commonalities with credit scoring.

Training data contains features and outcomes

1: fraud, 0: no fraud.

However, fraud detection poses a number of unique challenges.

Challenges of using ML for Fraud Detection

Data complexity and real-time processing.

Thousands of transactions per second.
Fraud must be detected (and blocked) in real time.

Class imbalance

Typically: 0.05% – 0.1% of transactions are fraudulent.
Trivial classifier (no fraud) is 99.9% – 99.95% accurate!

For credit scoring we would LOVE this level of accuracy.

Feature drift

Criminals mask fraudulent behavior in an evolving way.
Features associated to fraud one day, but not the next.

Class overlap

Criminals make legitimate transactions to trick the machine into missing fraudulent ones.

ML for Fraud Detection

There is an active body of research on using ML for fraud detection.

In addition to the above methods, people use

- Classification trees, ensemble learning, neural networks, Bayesian belief networks, hidden Markov models, etc.
- Unsupervised learning methods.