Industrial Engineering and Operations Research
COLUMBIA | ENGINEERING

Credit Card Risk Models: A Survey on Scoring and Limits

Research: Noah Dawang, Miao Wang, Dr Ali Hirsa, Satyan Malhotra

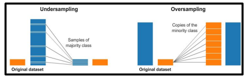
ask^2.ai

Abstract

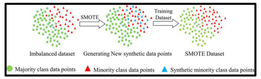
Credit cards are the most popular payment method in developed countries and their growth is extending internationally. On two datasets, one real and one simulated, we build and describe the entire pipeline of credit risk modelling, including exploratory data analysis, feature engineering, model assessment, interpretability, and assignment of limits

Resampling Methods for Imbalanced Data

Undersampling: Resample using the entire minority class, but only use a subset of the majority class to ensure balance in the data
Oversampling: Resample using the entire majority class and duplicate random rows of the minority class until balance is achieved

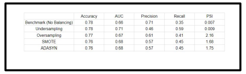


Synthetic Minority Oversampling Technique (SMOTE): Produce new synthetic instances of minority class by randomly selecting a minority class point, computing its k nearest neighbors, and produce a convex combination of the random point and a random nearest neighbor.


SMOTE
Training Dataset
Imbalanced dataset
Generating New synthetic data points
SMOTE Dataset
Majority class data points
Minority class data points
Synthetic minority class data points

ADASYN: A refinement of SMOTE that generates more new points around areas with a large majority class population.

Comparison of Results:



	Accuracy	AUC	Precision	Recall	PSI
Benchmark (No Balancing)	0.78	0.66	0.71	0.35	0.007
Undersampling	0.78	0.71	0.46	0.59	0.009
Oversampling	0.77	0.67	0.61	0.41	2.16
SMOTE	0.76	0.68	0.57	0.45	1.68
ADASYN	0.76	0.68	0.57	0.45	1.75

Modified SGT Embedding for Unstructured Transaction Data

Transaction data is a major component of behavioral models but it can come unstructured. We developed a new embedding algorithm, a modified version of sequence graph transform, to embed more information than could be done manually.

Algorithm 1 SGT Algorithm

```
    Input Alphabet set $V$ and a sequence $S$ composed of pairs $(t_i, e_i)$ where $t_i$ is a real number representing a
timestamp and $e_i$ comes from the alphabet set. $\kappa$, the hyperparameter
    Output A vector $x$ of length $|V|^2$ embedding $S$
  positions ← {}
  for $v \in V$ do
      positions($v$) ← {i: $e_i = v$}
  end for
  $W^{(0)}, W^{(\kappa)}, \Psi \leftarrow \mathbf{0}_{|V|\times|V|}, \mathbf{0}_{|V|\times|V|}, \mathbf{0}_{|V|\times|V|}$
  for $u \in V$ do
      $P_u \leftarrow$ positions($u$)
      for $v \in V$ do
          $P_v \leftarrow$ positions($v$)
          $C \leftarrow \{(i,j) | j > i \ \& \ i \in P_u, j \in P_v\}$
          $W^{(0)}_{uv} \leftarrow |C|$
          $E \leftarrow \{\exp(-\kappa(t_j - t_i)) | (i,j) \in C\}$
          $W^{(\kappa)}_{uv} \leftarrow \mathrm{sum}(E)$
          $\Psi_{uv} \leftarrow \left(\frac{W^{(\kappa)}_{uv}}{W^{(0)}_{uv}}\right)^{\frac{1}{\kappa}}$
      end for
  end for
  $x \leftarrow$ flatten($\Psi$)
```

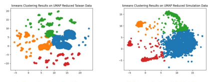
Dimension Reduction Algorithms

Dimension reduction is imperative for the visualization of data, and can be used in a pipeline for modelling.
Principal Component Analysis (PCA) computes the eigendecomposition of the covariance matrix and projects the data onto a small amount of eigenvectors
Robust Rolling Principal Component Analysis (R2-PCA) is a version of PCA adapted to time series that computes a time series of PCA analyses using a rolling window of covariance matrices
Kernel PCA first embeds to a higher dimension using a kernel function and computes a matrix of applying the kernel to each pair of points. Then, calculates the eigendecomposition and projects onto the eigenvectors similar to PCA.
Laplacian Eigenmaps constructs a graph structure in the data and computes the eigendecomposition of the graph laplacian matrix. Associates each point to the first few components of an eigenvector, starting with the second smallest eigenvalue.
t-distributed Stochastic Neighbor Embedding (t-SNE) constructs a graph from the data using probability distance and finds a low dimensional representation that minimizes the KL divergence between the probability distance distributions of the low dimensional and high dimensional representations
Uniform Manifold Approximation and Projection (UMAP) constructs a graph structure similar to t-SNE using more deliberate methods and computes the low dimensional distance using a more refined and tested formula
Autoencoder trains a neural network to learn itself with a low dimensional bottleneck in the middle. The low dimensional layer is considered the dimension reduction.

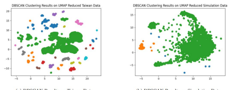
Clustering Algorithms

The objective of clustering is to accurately segment customers based on their demographics and behavior. Ideally, models can be specialized for different clusters

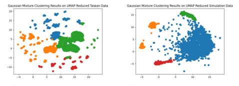
k-means is a linear clustering algorithm that finds k centroids in the data that classify each point by nearest distance


(a) k-means Results on Taiwan Data (b) k-means Results on Simulation Data

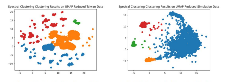
Robust Rolling k-means (R2-kmeans) is an adaptation of k-means for time series. It uses a rolling window of past centroids as memory and uses the previous window's centroids as initialization
Density-Based Spatial Clustering of Applications with Noise (DBSCAN) finds clusters using density of points. Hyperparameters minPts and ε are used to measure density around a point. A core point is one where there are at least minPts in a ball of radius ε and a border point are all other points. DBSCAN finds neighborhoods of core points and assigns them to the same cluster recursively until all points are assigned a cluster


(a) DBSCAN Results on Taiwan Data (b) DBSCAN Results on Simulation Data

Gaussian Mixture fits a mixture of normal distributions to the data using the expectation maximization algorithm. Points are assigned to a cluster based on a posterior probability calculation


(a) Gaussian Mixture Results on Taiwan Data (b) Gaussian Mixture Results on Simulation Data

Spectral Clustering constructs a graph structure from the data and calculates the eigendecomposition of the graph laplacian matrix. It then clusters the eigenvectors using another clustering algorithm, most commonly k-means

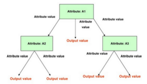

(a) Spectral Clustering Results on Taiwan Data (b) Spectral Clustering Results on Simulation Data

Supervised Learning Models

Supervised models learn the risk of default by learning a function pd(x) that estimates the probability of a customer with features x to default.
Logistic Regression fits a model with pd(x) taking the following parametric form

$$pd(x) = \frac{1}{1+e^{-x\beta}}$$

Decision tree models use as their building blocks flowcharts that classify based on thresholds of features.



Adaboost fits an ensemble of decision trees sequentially by reweighting each previously misclassified datapoint using the following formula

$$w_i^{(n)} = w_i^{(n-1)} \exp\left(\alpha_{n-1} I(f_{n-1}(x_i) \neq y_i)\right)$$
$$\alpha_{n-1} = \ln\left(\frac{1-\epsilon_{n-1}}{\epsilon_{n-1}}\right)$$

Gradient Boosting fits an ensemble of decision trees sequentially by training each successive tree on the error of the previous iteration
Random Forest fits an ensemble of decision trees in parallel using a subset of features and/or data points in each tree
Support Vector Machines (SVM) use a linear classifier based on minimizing the hinge loss

$$\text{Hinge Loss} = ||\beta||^2 + C\sum_{i=1}^{n} \max(1 - y_i\hat{y}_i, 0)$$

The classifier can be made nonlinear by embedding in a higher dimensional space using a kernel function
Neural Networks use layers called neurons. Neurons are propagated with an affine transformation and a nonlinear activation function. There are many types of neural networks which can be applied to different types of data
Multilayer Perceptrons (MLP) have multiple hidden layers of fully connected layers
Recurrent Neural Networks use previous iterations of neurons fit while training
Convolutional Neural Networks are effective for time series and image data, and use convolutions instead of matrix multiplication for the linear transformations
Transformers are effective for text data and use self attention layers in sequence with normal fully connected layers

Credit Limit Models

Static Limit Models initialize a new customer's credit limit based on their application score and other attributes. We formulate the model as follows:

1. X: Data matrix of features. These are the holding features that both credit card and non credit card consumers have
2. X_i: Data vector of the ith consumer
3. $pd(X_i)$: The output of application scoring model on customer i, given as the probability of default
4. $pd(X_i, L_i)$: The probability of default on consumer X_i, given that we assign them a credit limit L_i.
5. $U(X_i, L_i)$: The net credit usage of customer i. This can be calculated from the data in a few ways.
6. $D(X_i)$: The net demand for credit of customer i. This is how much credit a customer desires and may exceed the limit they are assigned
7. $EP_i(L)$: The expected profit from the ith customer given we assign a credit limit of L to them
8. r: The APR for a credit card. In our case, it is approximately constant between 22% and 27%
9. L_i^*: The optimal limit for customer i

We do an expected profit driven limit assignment strategy, where we assign a customer's limit in order to maximize the expected profit

$$EP_i(L) = \mathbb{E}[U(X_i, L)](-pd(X_i, L) + (1 - pd(X_i, L))(1 + r))$$
$$L_i^* = argmax_L EP_i(L)$$

Accelerated Failure Time (AFT) models U(X,L) using the customer's net demand for credit D(X) with the relationship being U(X,L) = min(D(X),L). The AFT model is

$$\log D(X) = f(X) + \sigma\epsilon$$

Where f(X) is a predictor on X (linear, decision tree, neural network etc) and ε has a predefined error distribution with probability density ϕ(z) and cumulative density Φ(z). Due to the censored nature of the data, we fit using maximum likelihood

$$\log \mathcal{L}(\sigma, \theta) = \sum_{y_i < L_i} \log\left(\frac{1}{\sigma}\phi\left(\frac{y_i - f(X_i)}{\sigma}\right)\right) + \sum_{y_i = L_i} \log\left(\Phi\left(\frac{f(X_i) - L_i}{\sigma}\right)\right)$$

Dynamic Limit Models are responsible for credit limit adjustment due to changes in customer behavior or demographic information. We formulate the model using a **Markov Decision Process (MDP)**. The MDP models the state as the net end of period balance and the behavioral score. The objective is to maximize the expected discounted future profit. We model the transition matrix between states P as a function of previous states and our choice of limit. Then,, we discretize probability of default into score buckets S and net credit balance into buckets B. The expected profit at time t is

$$EP_t(B_{t-1}, S_{t-1}, L_t) = \sum_{B_t, S_t} P(B_t, S_t | B_{t-1}, S_{t-1}, L_t) B_t(-pd(S_t) + (1 - pd(S_t))(1 + r))$$

The [illegible] a the dynamic program

$$V(B_{t-1}, S_{t-1}) = \max_{L_t, B_{t-1}, S_{t-1}} \{EP_t(B_{t-1}, S_{t-1}, L_t) + \beta \sum_{B_t, S_t} P(B_t, S_t | B_{t-1}, S_{t-1}, L_t) V(B_t, S_t)\}$$