Classification Modeling with Imbalanced Classes

Abstract

Imbalanced classification is a supervised learning problem where one class outnumbers other class by a large proportion. This problem is faced more frequently in binary classification problems than multi-level classification problems. This imbalanced distribution of classes is quite commonly faced in business problems these days.

The standard statistical learning algorithms like Maximum Likelihood etc. fare quite well in attaining an almost universal accuracy. But such algorithms usually lead to a bugbear called **Accuracy Paradox**. The accuracy paradox for predictive analytics states that predictive models with a given level of accuracy may have greater predictive power than models with higher accuracy. Thus, a model with higher accuracy makes the analyst to have a false sense of success, but inferences or generalization through that model takes an unwarranted hit.

I hope to come up with some algorithm or suggest some other way which successfully facilitates the pure classification of data with imbalanced classes.

Dataset

The Dataset used throughout this paper is the "Kaggle Credit Card Fraud Detection Dataset." The datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Techniques

- **Logistic Regression**
- **Weighted Logistic Regression**
- **Re-sampling techniques**
    - **Under-sampling the majority class(es)**
    - **Over-sampling**
    - **Over-sampling followed by under-sampling(SMOTE)**
    - **Case Control Sampling**
- **Cost Sensitive Learning**
    - **Wrapper based cost sensitive learning**
    - **C 5.0 Decision tree with cost**
- **Firth Logistic Regression (penalized likelihood)**
- **Exact Logistic Regression**
- **Novelty Detection**
    - **Support Vector Machines**

Performance Metrics

Classification is about predicting class labels given input data. In binary classification, there are two possible output classes. In multi-class classification, there are more than two possible classes. An example of binary classification is spam detection, where the input data could be the email text and metadata (sender, sending time), and the output label is either "spam" or "not spam." Sometimes, people use generic names for the two classes: "positive" and "negative," or "class 1" and "class 0."

Performance Metrics used in our research are:

Accuracy:

Accuracy simply measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions (the number of test data points).

Precision and Recall:

Precision and recall are actually two metrics. But they are often used together. Precision answers the question: Out of the items that the classifier predicted to be true, how many are actually true? Whereas, recall answers the question: Out of all the items that are true, how many are found to be true by the classifier?

The precision score quantifies the ability of a classifier to not label a negative example as positive. The precision score can be interpreted as the probability that a positive prediction made by the classifier is positive. The score is in the range [0,1] with 0 being the worst, and 1 being perfect.

F1 Score:

The F1-score is a single metric that combines both precision and recall via their harmonic mean. The score lies in the range [0,1] with 1 being ideal and 0 being the worst. Unlike the arithmetic mean, the harmonic mean tends toward the smaller of the two elements. Hence the F1 score will be small if either precision or recall is small.

Kappa score:

In essence, the kappa statistic is a measure of how closely the instances classified by the *machine learning classifier* matched the data labeled as *ground truth*, controlling for the accuracy of a random classifier as measured by the expected accuracy.

Kappa = (observed accuracy - expected accuracy)/(1 - expected accuracy)

Observed Accuracy is simply the number of instances that were classified correctly throughout the entire confusion matrix

Expected Accuracy. This value is defined as the accuracy that any random classifier would be expected to achieve based on the confusion matrix. The Expected Accuracy is directly related to the number of instances of each class along with the number of instances that the *machine learning classifier* agreed with the *ground truth* label.

Receiver Operating Characteristic Curve (ROC Curve):

In statistics, a receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its prediction **threshold** is varied. The ROC curve provides nuanced details about the behavior of the classifier. The curve is created by plotting the **true positive rate** (TPR) against the **false positive rate** (FPR = 1-sensitivity = Recall) at various threshold settings.

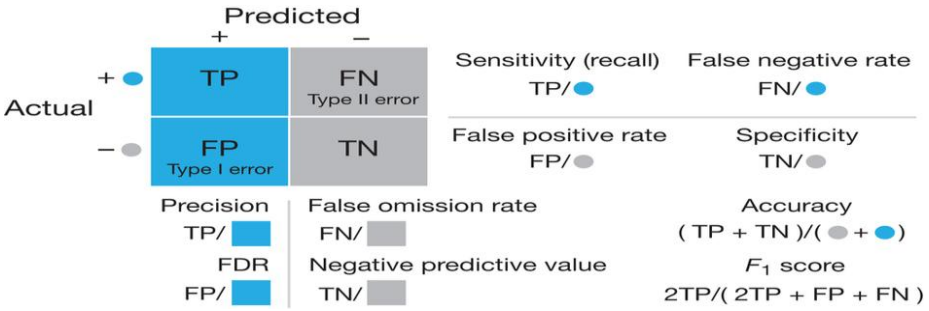
The ROC curve shows the sensitivity of the classifier by plotting the rate of true positives to the rate of false positives. In other words, it shows you how many correct positive classifications can be gained as you allow for more and more false positives. The perfect classifier that makes no mistakes would hit a true positive rate of 100% immediately, without incurring any false positives. This almost never happens in practice.

A point on the ROC curve, say with **Sensitivity = 0 .98041 and Specificity = 0.9011(FPR = .10)** in our credit card fraud data will imply that, 98% of innocent people will be correctly classified as innocent and 10% of guilty people will be classified as innocent. We aim to get high sensitivity with low FPR.

Area under the curve (AUC):

This Statistic measures the area under the ROC Curve and gives us an idea about the performance of the model. As it is dependent on the ROC curve, better the ROC curve, better the AUC score. AUC ranges between 0 and 1 and a good score is considered to be above 0 .75.

A Brief of the Performance Metrics



Logistic Regression

Logistic Regression, also known as *Logit Regression* or *Logit Model*, is a mathematical model used in statistics to estimate (guess) the probability of an event occurring having been given some previous data. Logistic Regression works with binary data, where either the event happens (1) or the event does not happen (0). So given some feature x it tries to find out whether some event y happens or not. So y can either be 0 or 1. In the case where the event happens, y is given the value 1. If the event does not happen, then y is given the value of 0. For example, if y represents whether a sports teams wins a match, then y will be 1 if they win the match or y will be 0 if they do not.

Logistic regression uses the concept of odds ratios to calculate the probability. This is defined as the ratio of the odds – or probability – of an event happening to it not happening. For example, the probability of a sports team to win a certain match might be 0.75. The odds for that team to lose would be 1 – 0.75 = 0.25. The odds ratio for that team winning would be 0.75/0.25 = 3. This can be said as the odds of the team winning are 3 to 1 on.

The odds ratio can then be defined as:

$$odds = \frac{p}{1-p}$$

The natural logarithm of the odds ratio is then taken in order to create the logistic equation. The new equation is known as the logit:

$$\log(odds) = logit(p) = \ln\left(\frac{p}{1-p}\right)$$

In Logistic regression the Logit of the probability is said to be linear with respect to x, so the logit becomes:

$$\ln\left(\frac{p}{1-p}\right) = a + bX$$

Using the two equations together then gives the following:

$$\frac{p}{1-p} = e^{a+bX}$$

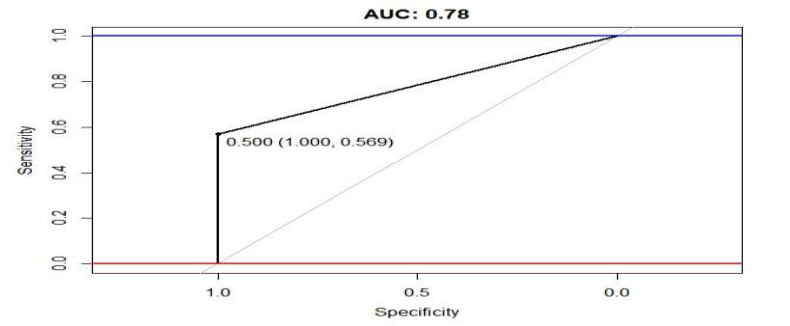
This then leads to the probability:

$$p = \frac{e^{a+bX}}{(1+e^{a+bX})}$$

This final equation is the logistic curve for Logistic regression. It models the non-linear relationship between x and y with an 'S'-like curve for the probabilities.

Performance Metrics:

(ROC Curve)



Techniques	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
Simple logistic Regression	0.9991	0.999234	**0.9998**	**0.9995**	**0.6804**	0.9998	0.5692	0.7845

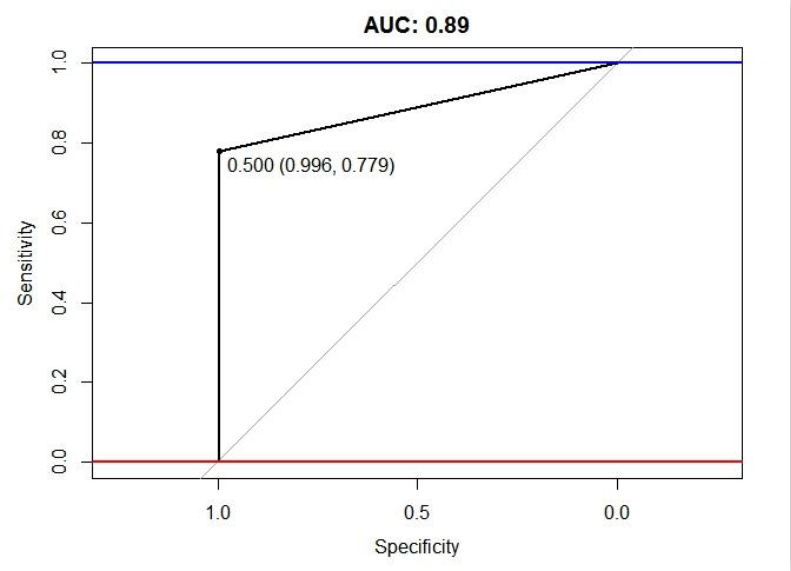
Advantages	Disadvantages
Computation is easy and packages are widely available in most software. It also includes "predict" functionality for validating models.	Requires large dataset to make meaningful and stable predictions. At least 50 data points per predictor is required for stable results.
It is robust: the independent variables don't have to be normally distributed, or have equal variance in each group	Logistic regression aims to increase accuracy and thus struggles from accuracy paradox in case of imbalance datasets.
No linear relationship between the independent variable and dependant variable has to be assumed.	Logistic regression requires that each data point be independent of all other data points. If observations are related to one another, then the model will tend to overweight the significance of those observations.
Logistic Regression Can handle nonlinear effect, interaction effect and power terms	Logistic regression attempts to predict outcomes based on a set of independent variables, but logit models are vulnerable to overconfidence. That is, the models can appear to have more predictive power than they actually do as a result of sampling bias.

Weighted Logistic Regression

This classifier is similar to that in logistic regression, except, it gives each observation a pre-defined weight variable. This causes the classifier to give more emphasis on observations that have higher weights. In our case, we use the amount of each transaction as the weight. Intuitively, we would agree with this as predicting a high-value transaction correctly is more valuable financially as compared to predicting a low value transaction correctly. This method may not improve performance metrics but it does help us predict financially important transactions correctly.

Performance Metrics:

(ROC Curve)



Technique	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
Weighted Logistic Regression	0.9961	0.9996	0.9964	0.998	0.4141	0.9965	0.7787	0.8876

We have been able to gain AUC, but we can see that Kappa coefficient has dropped significantly. Though the increased AUC is a good sign, when compared with the reduced Kappa, it does not seem very optimal.

1) Sampling Methods

Generally, these methods aim to modify an imbalanced data into balanced distribution using some mechanism. The modification occurs by altering the size of original data set and provides the same proportion of balance.

These methods have acquired higher importance after many researchers have proved that balanced data results in improved overall classification performance compared to an imbalanced data set. Hence, it's important to learn them.

Below are the methods used to treat imbalanced datasets:

1. Undersampling
2. Oversampling
3. Case-Control Sampling
4. Synthetic Data Generation (SMOTE)

1.1) Undersampling

This method works with majority class. It reduces the number of observations from majority class to make the data set balanced. This method is best to use when the data set is huge and reducing the number of training samples helps to improve run time and storage troubles. Undersampling method *randomly* chooses observations from majority class which are eliminated until the data set gets balanced.

Problem with undersampling methods? Removing observations may cause the training data to lose important information pertaining to majority class.

1.2) Oversampling

This method works with minority class. It replicates the observations from minority class to balance the data. It is also known as upsampling. Similar to undersampling, this method also can be divided into two types. oversampling balances the data by randomly oversampling the minority class.

An advantage of using this method is that it leads to no information loss. The disadvantage of using this method is that, since oversampling simply adds replicated observations in original data set, it ends up adding multiple observations of several types, thus leading to overfitting. Although, the training accuracy of such data set will be high, but the accuracy on unseen data will be worse.

1.3) Case-Control Sampling

A case-control study is a type of observational study in which two existing groups differing in outcome are identified and compared on the basis of some supposed causal attribute. Case-control sampling is widely used in epidemiology and medical studies but this study finds application in our case as well. The study compares people who had fraudulent transactions or outcome of interest (cases) with patients who had legitimate transactions (controls) and looks back retrospectively to compare how frequently the exposure to a risk factor is present in each group to determine the relationship between the risk factor and the fraud.

Case control studies are observational because no intervention is attempted and no attempt is made to alter the course of the fraud. The goal is to retrospectively determine the exposure to the risk factor of interest from each of the two groups of individuals: cases and controls.

It can be seen as a derivative of undersampling but with certain controls. We keep all the Cases in our data and undersample our data to get a balance of about 85:15. Studies have shown that this ratio normally gives us stable results and it had also been proven further (SMOTE) that this ratio fares better.We have further seen that implementing case control sampling entails some bias in our

estimate of the intercept and thus a correction factor is used in order to correct the bias. The predictions are then made based on this corrected model.

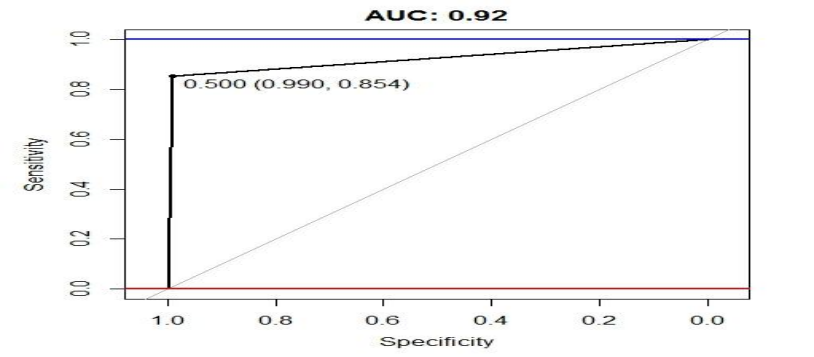
The correction factor is:

$$\hat{B}_0^* = \hat{B}_0 + \log\left(\frac{\pi}{1-\pi}\right) - \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right)$$

$$\pi = Original\ Ratio$$

$$\hat{\pi} = New\ Ratio$$

Performance Metrics



Techniques	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
Case Control Sampling	0.9901	0.999744	0.990338	0.99502	0.2273	0.9903	0.8537	0.922

Advantages	Disadvantages
Good for studying rare conditions or diseases as it utilizes all the rare observations and makes use of all the information in the rare class.	It is subject to selection bias. Though the correction factor stated gives us some relief from this drawback.
It is robust: It can be used with any base learner, such as Logistic Regression, C4.5, Naïve Bayes classifier etc., to address the class imbalance problem.	Though the study uses all of the rare data points, we lose out on large amount of data in the common class.
Lets you simultaneously look at multiple risk factors	
Useful as initial studies to establish an association	

1.4) SMOTE: Synthetic Minority Over-sampling Technique

SMOTE is a well-known algorithm to fight the problem of imbalance classes. The general idea of this method is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset.

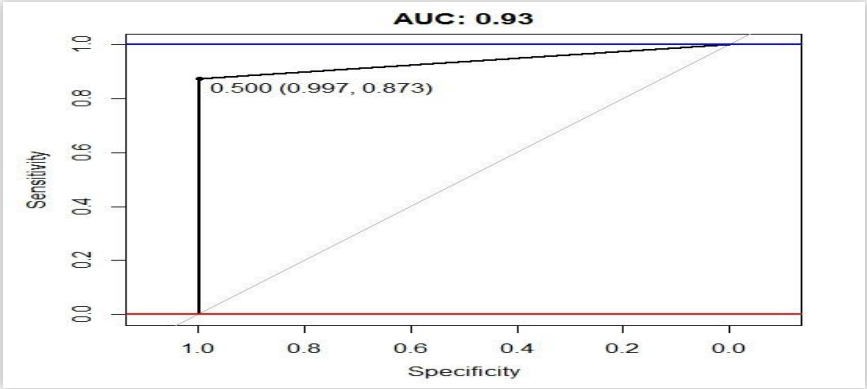
The parameters perc.over and perc.under control the amount of over-sampling of the minority class and under-sampling of the majority classes, respectively. perc.over will typically be a number above 1. With this type of values, for each case in the original data set belonging to the minority class, perc.over new examples of that class will be created. If perc.over is a value below 1 then a single case will be generated for a randomly selected proportion (given by perc.over) of the cases belonging to the minority class on the original data set. The parameter perc.under controls the proportion of cases of the majority class that will be randomly selected for the final "balanced" data set. This proportion is calculated with respect to the number of newly generated minority class cases. For instance, if 200 new examples were generated for the minority class, a value of perc.under of 100 will randomly select exactly 200 cases belonging to the majority classes from the original data set to belong to the final data set. Values above 100 will select more examples from the majority classes.

The parameter k controls the way the new examples are created. For each currently existing minority class example X new examples will be created (this is controlled by the parameter perc.over as mentioned above). These examples will be generated by using the information from the k nearest neighbours of each example of the minority class. The parameter k controls how many of these neighbours are used

SMOTE Algorithm

Consider a sample (6,4)an d let (4,3)b e its nearest neighbor.
(6,4)is the sample for which k-nearest neighbors are being identified.
(4,3)is one of its k-nearest neighbors.
Let:
$f_{11} = 6\ f_{21} = 4\ f_{21} - f_{11} = -2$
$f_{12} = 4\ f_{22} = 3\ f_{22} - f_{12} = -1$
The new samples will be generated as
$(f_1', f_2') = (6,4) + rand(0-1) * (-2,-1)$

Performance Metrics



Techniques	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
SMOTE(60:40)	0.9803	0.99982	0.98041	0.99002	0.1368	0.98041	0.90119	0.9408
SMOTE(70:30)	0.9896	0.999794	0.9898	0.99477	0.2301	0.9898	0.8854	0.9305
SMOTE(75:25)	0.9911	0.999794	0.9898	0.99477	0.258	0.9913	0.8775	0.9289
SMOTE(86:14)	0.9969	0.999732	0.99718	0.99845	0.4935	0.9972	0.8498	0.9235
SMOTE(86:14)+WLR	0.9956	0.999746	0.99589	0.99782	0.4102	0.9959	0.8577	0.9305
SMOTE(86:14)+GBM	**0.9966**	**0.999788**	**0.99676**	**0.99827**	**0.4554**	**0.9968**	**0.8729**	**0.9348**

Advantages	Disadvantages
Computation is easy and packages are widely available in most software. It also includes "predict" functionality for validating models.	Strict assumption that the local space between any two positive instances is positive or belongs to the minority class, which may not always be true in the case when the training data is not linearly separable.
It is robust: It can be used with any base learner, such as Logistic Regression, C4.5, Naïve Bayes classifier etc., to address the class imbalance problem.	While in most cases SMOTE seems beneficial with low-dimensional data, it does not attenuate the bias towards the classification in the majority class for most classifiers when data are high-dimensional, and it is less effective than random undersampling.
Due to synthetic samples, SMOTE method avoids over-fitting largely and achieves a good performance in the imbalanced data classification problem.	SMOTE introduces correlation between some samples, but not between variables. SMOTE can be problematic for the classifiers that assume independence among samples, as for example penalized logistic regression or discriminant analysis methods.
Many highly skewed data sets are enormous and the size of the training set must be reduced in order for learning to be feasible. In this case, SMOTE seems to be a reasonable, and valid, strategy.	The overall expected value of the SMOTE-augmented minority class is equal to the expected value of the original minority class, while its variance is smaller. Therefore, SMOTE has little impact on the classifiers that base their classification rules on class-specific mean values and overall variances (as DLDA), while it has some (harmful) impact on the classifiers that use class-specific variances (as DQDA), because they use biased estimates

2) Cost Sensitive Learning

It is another commonly used method to handle classification problems with imbalanced data. It's an interesting method. In simple words, this method evaluates the cost associated with misclassifying observations.

It does not create balanced data distribution. Instead, it highlights the imbalanced learning problem by using cost matrices which describes the cost for misclassification in a particular scenario. Recent researches have shown that cost sensitive learning have many a times outperformed sampling methods. Therefore, this method provides likely alternative to sampling methods.

Let's understand it using an example: In the credit card data that we have, we are interested in knowing which transactions are fraudulent and which ones are legitimate. A fraudulent transaction is labeled as a positive class, and legitimate transactions is labeled as a negative class. The problem is to identify the fraudulent transactions. Now, understanding the cost matrix, there is no cost associated with identifying a fraudulent transaction as positive and a legitimate transaction as negative. But, the cost associated with identifying a fraudulent transaction as negative(False Negative) is higher than identifying a legitimate transaction as positive(False Positive).

Cost Matrix is similar of confusion matrix. It's just; we are here more concerned about false positives and false negatives (shown below). There is no cost penalty associated with True Positive and True Negatives as they are correctly identified.

Cost Matrix	Predicted Positive	Predicted Negative
Actual Positive	$C_{00} = 0$	$C_{01} = C(FN)$
Actual negative	$C_{10} = C(FP)$	$C_{11} = 0$

The goal of this method is to choose a classifier with lowest total cost.

$$\text{Total Cost} = C(FN) \times FN + C(FP) \times FP$$

1. FN is the number of positive observations wrongly predicted
2. FP is the number of negative examples wrongly predicted
3. C(FN) and C(FP) corresponds to the costs associated with False Negative and False Positive respectively. Remember, C(FN) > C(FP).

Cost Sensitive learning can be implemented in the following ways:

1) Using Predicted Probabilities
2) By Rebalancing

Using predicted probabilities

Initially, we train a classifier on a training dataset and then validate it on a testing dataset. The classifier, in our case logistic regression, will give probabilities of each observation. Normally, a cutoff point of 0.5 will be set, with all probabilities above 0.5 classified as 1, and others as 0. In cost sensitive learning, we aim to change these probabilities using the cost matrix.

Given a specification of costs for correct and incorrect predictions, an example should be predicted to have the class that leads to the lowest expected cost, where the expectation is computed using the conditional probability of each class given the example. Mathematically, let the (i, j) entry in a cost matrix C be the cost of predicting class i when the true class is j . If i = j then the prediction is correct, while if i $\neq$ j the prediction is incorrect. The optimal prediction for an example x is the class i that minimizes:

$$L(x,i) = \sum_j P(j|x) * C(i,j)$$

For each i , L(x,i) is a sum over the alternative possibilities for the true class of x. In this framework, the role of a learning algorithm is to produce a classifier that for any example x can estimate the probability P(j|x) of each class j being the true class of x. For an example x, making the prediction i means acting as if i is the true class of x. The essence of cost-sensitive decision-making is that it can be optimal to act as if one class is true even when some other class is more probable. For example, it can be rational not to approve a large credit card transaction even if the transaction is most likely legitimate.

Optimal Decisions

In the two-class case, the optimal prediction is class 1 if and only if the expected cost of this prediction is less than or equal to the expected cost of predicting class 0, i.e. if and only if

$$P(j=0|x)C10 + P(j=1|x)C11 \leq P(j=0)C00 + P(j=1|x)C01$$

Which is equivalent to:

$$(1-P)C10 + PC11 \leq (1-P)C00 + PC01$$

Given $p = P(j=1|x)$, If this inequality is in fact an equality, then predicting either class is optimal. The threshold for making optimal decisions is P* such that

$$(1-P^*)C10 + P^*C11 = (1-P^*)C00 + P^*C01$$

Rearranging the equation for P* leads to the solution

$$P^* = \frac{(C10-C00)}{(C10-C00+C01-C11)}$$

Cost Sensitivity by Rebalancing

Standard learning algorithms are designed to yield classifiers that maximize accuracy. In the two-class case, these classifiers implicitly make decisions based on the probability threshold 0.5. The conclusion of the previous section was that we need a classifier that given an example x, says whether or not $P(j=1|x) \geq P^*$ for some target threshold P^* that in general is different from 0.5. How can a standard learning algorithm be made to produce a classifier that makes decisions based on a general P^*?

The most common method of achieving this objective is to rebalance the training set given to the learning algorithm, i.e. to change the proportion of positive and negative training examples in the training set. Although rebalancing is a common idea, the general formula for how to do it correctly has not been published. The following theorem provides this formula.

Theorem: To make a target probability threshold P^* correspond to a given probability threshold P_0, the number of negative examples in the training set should be multiplied by:

$$\frac{P^*}{1-P^*} * \frac{1-P_0}{P_0}$$

If a learning algorithm can use weights on training examples, then the weight of each negative example can be set to the factor given by the theorem. Otherwise, we must do oversampling or undersampling.

Performance Diagnostics

Techniques	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
Cost Sensitive learning(0,2,6,0)	0.999206	0.999	0.999	0.999	0.7312			0.975
C 5.0 decision Tree	0.9993	0.999515	0.99983	0.99967	0.7819	0.9998	0.7076	0.8537
C 5.0 + Cost(0,2,6,0)	0.9993	0.999571	0.99977	0.99967	0.788	0.9998	0.7415	0.8706

Advantages	Disadvantages
Helps make optimal decisions; the essence of cost-sensitive decision-making is that it can be optimal to act as if one class is true even when some other class is more probable. For example, it can be rational not to approve a large credit card transaction even if the transaction is most likely legitimate.	There are not cost-sensitive implementations of all learning algorithms and therefore a wrapper-based approach using sampling is the only option
The major advantages of these methods are the simplicity without any change in algorithms and the applicability to any existing error-based classification learning algorithms. They generate better models in terms of overall costs, compared to the models derived from original sample data	Misclassification Costs are often unknown or are difficult to determine, thus the cost used in the model, though giving good results may turn out to be infeasible.

3) Exact and Firth logistic regression

Exact Logistic Regression

Exact logistic regression is used to model binary outcome variables in which the log odds of the outcome is modeled as a linear combination of the predictors. It is used when the sample size is too small for a regular logistic regression (which uses the standard maximum-likelihood-based estimator) and/or when some of the cells formed by the outcome and categorical predictor variable have no observations. The estimates given by exact logistic regression do not depend on asymptotic results. It is also used to counter the problem of separation in the dataset.

It uses MCMC sampling. Markov chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. The state of the chain after a number of steps is then used as a sample of the desired distribution. The quality of the sample improves as a function of the number of steps.

It is thus computationally very intensive and could not be successfully conducted on our dataset.

Firth Logistic Regression or Penalized likelihood regression

The problem is not specifically the rarity of events, but rather the possibility of a small number of cases on the rarer of the two outcomes. If you have a sample size of 1000 but only 20 events, you have a problem. If you have a sample size of 10,000 with 200 events, you may be OK. If your sample has 100,000 cases with 2000 events, you're golden. There's nothing wrong with the logistic model in such cases. The problem is that maximum likelihood estimation of the logistic model is well-known to suffer from small-sample bias. And the degree of bias is strongly dependent on the number of cases in the less frequent of the two categories. So even with a sample size of 100,000, if there are only 20 events in the sample, you may have substantial bias.

The Solution? The Firth method, after its inventor, penalized likelihood is a general approach to reducing small-sample bias in maximum likelihood estimation. In the case of logistic regression, penalized likelihood also has the attraction of producing finite, consistent estimates of regression parameters when the maximum likelihood estimates do not even exist because of complete or quasi-complete separation.

Unlike exact logistic regression, penalized likelihood takes almost no additional computing time compared to conventional maximum likelihood. In fact, a case could be made for always using penalized likelihood rather than conventional maximum likelihood for logistic regression, regardless of the sample size.

FIRTH'S MODIFIED SCORE PROCEDURE

Maximum likelihood estimates of regression parameters $\beta_r (r = 1,2, \dots k)$ are obtained as solutions to the score equations $\frac{\partial \mathrm{LogL}}{\partial \beta_r} \equiv U(\beta_r) = 0$ where L is the likelihood function.

In order to reduce the small sample bias of these estimates Firth [19] suggested basing estimation on modified score equations:

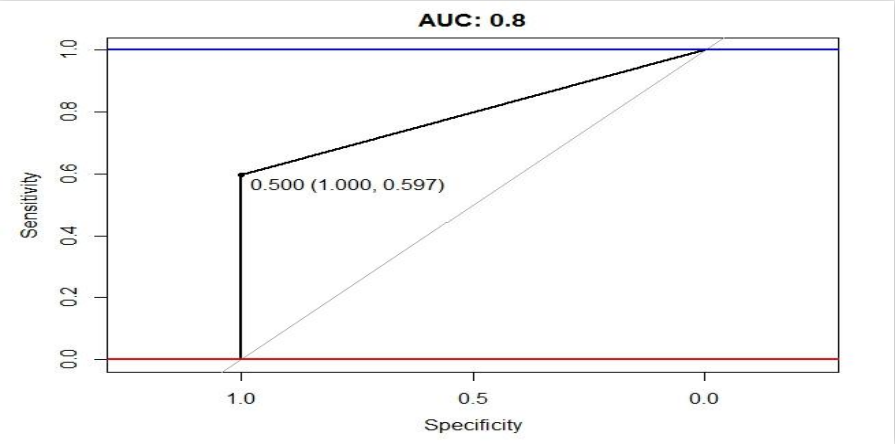
$$U(\beta_r) * \equiv U(\beta_r) + \frac{1}{2}\mathrm{trace}[I(\beta)^{-1}\left\{\frac{\partial I(\beta)}{\partial \beta_r}\right\}] = 0 \ , (r = 1,2, \dots k)$$

Where $I(\beta)^{-1}$ is the inverse of the information matrix evaluated at β. The modified score function $U(\beta_r) *$is related to the penalized log-likelihood and likelihood functions,$\log L(\beta) * = \log L(\beta) + \frac{1}{2}\log |I(\beta)|$ and $L(\beta) * = L(\beta)|I(\beta)|^{1/2}$, respectively.

The penalty function $|I(\beta)|^{1/2}$is known as Jeffreys invariant prior. It's influence is asymptotically negligible. By using this modification Firth [19] showed that the $O(n^{-1})$ bias of maximum likelihood estimates $\widehat{\beta}$ is removed.

Performance Metrics

Technique	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
Firth Logistic Regression	0.9991	0.999332	0.9997468	0.9995394	0.6824	0.9997	0.5975	0.813



Advantages	Disadvantages
It alleviates the problem of the curse of dimensionality in very large databases	The firth logistic package 'flogit' does not have in built predict functionality. Thus we have to use 'brglm' as an approximation.
It deals with missing values implicitly. It uses the connectivity property of points to deal efficiently with missing values.	Computationally Intensive, may take long time to compute if the dataset is sufficiently large. Thus it is more costly to implement
It has the ability to adapt to non-stationary sources of data. The method has been successfully tested using network intrusion and fraud detection.	The results produced in our case are not very different from that of SMOTE and simple logistic regression which is computationally easy.

4) Novelty Detection

Anomaly detection (also outlier detection) is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset. Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions.

Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then testing the likelihood of a test instance to be generated by the learnt model.

Anomaly detection is applicable in a variety of domains, such as intrusion detection, fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting Eco-system disturbances. It is often used in preprocessing to remove anomalous data from the dataset. In supervised learning, removing the anomalous data from the dataset often results in a statistically significant increase in accuracy.

There are numerous ways to do anomaly detection in the data. We specifically focus on one major technique:

1) One-Class Support Vector Machine

Other techniques are included in the references and are to be used for further research.

4.1) One-Class Support Vector Machine

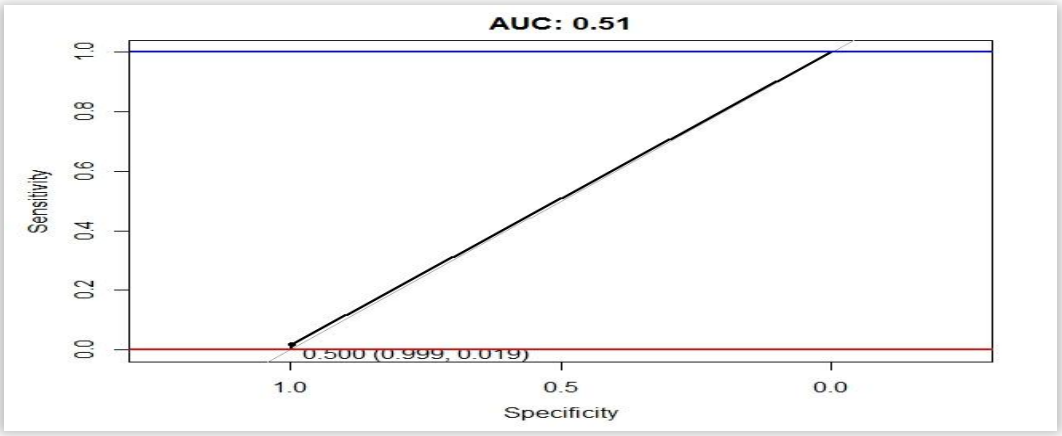
Support vector machines (SVMs) are supervised learning models that analyze data and recognize patterns, and that can be used for both classification and regression tasks.

Typically, the SVM algorithm is given a set of training examples labeled as belonging to one of two classes. The SVM algorithm represents the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to one category or another, based on which side of the gap they fall on.
Sometimes oversampling is used to replicate the existing samples so that you can create a two-class model, but it is impossible to predict all the new patterns of fraud or system faults from limited examples. Moreover, collection of even limited examples can be expensive.

Therefore, in one-class SVM, the support vector model is trained on data that has only one class, which is the “normal” class. It infers the properties of normal cases and from these properties can predict which examples are unlike the normal examples. This is useful for anomaly detection because the scarcity of training examples is what defines anomalies: that is, typically there are very few examples of the network intrusion, fraud, or other anomalous behavior.

The best parameters to be used depend on the dataset and would require a hyperparameter search in order to find the best results. Below are the results from one such combination of parameters.

Performance Metrics:



Technique	Accuracy	Precision	Recall	F-score	Kappa	Sensitivity	Specificity	AUC
SVM anomaly	0.9479	0.9992145	0.948548	0.9732223	0.0343	0.94855	0.57769	0.5093

Advantages	Disadvantages
It alleviates the problem of the curse of dimensionality in very large databases	assumes that outliers are far away from the "normal" points in data space
It deals with missing values implicitly. It uses the connectivity property of points to deal efficiently with missing values.	Computationally Intensive, may take long time to compute if the dataset is sufficiently large. Thus it is more costly to implement
It has the ability to adapt to non-stationary sources of data. The method has been successfully tested using network intrusion and fraud detection.	The results produced in our case are not promising. The AUC value is close to 0.5 i.e it may as well be random.