भारतीय प्रबंध संस्थान बेंगलूर
INDIAN INSTITUTE OF MANAGEMENT
BANGALORE

Using Machine Learning Algorithms to Detect Earnings Manipulation

Context

1. Earnings management occurs when managers use their judgment in financial reporting and in structuring transactions to alter financial reports to mislead stakeholders about the underlying economic performance of the company

2. As on December 2016, the total number of listed companies in Indian stock exchange was approximately 5622.

3. Securities and Exchange Board of India (SEBI) reported that approximately 3.14% of the Indian companies are involved in earnings manipulation.

Literature Review

भारतीय प्रबंध संस्थान बेंगलूर
INDIAN INSTITUTE OF MANAGEMENT BANGALORE

Beneish Model	Data Used	Outcome
The earliest work carried out on predicting the earnings manipulation was done by Beneish (1997, 1999). Devised an earnings manipulation model based on Probit regression.	The model used eight financial ratios to build M – Score (manipulation score) to identify companies who are likely to have manipulated financial books.	Model looked into the data reported by U.S companies. The data used was not imbalanced and thus probit regression model was able to give a better accuracy in classification.

Financial Ratios

भारतीय प्रबंध संस्थान बेंगलूर
INDIAN INSTITUTE OF MANAGEMENT BANGALORE

Days Sales to Receivables Index (DSRI)

$$DSRI = \frac{\frac{Receivable_{(t)}}{Sales_{(t)}}}{\frac{Receivable_{(t-1)}}{Sales_{(t-1)}}}$$

DSRI greater than 1 implies revenue inflation

Gross Margin Index (GMI)

$$GMI = \frac{\frac{Sales_{(t-1)} - Cost\ of\ Goods\ Sold_{(t-1)}}{Sales_{(t-1)}}}{\frac{Sales_{(t)} - Cost\ of\ Goods\ Sold_{(t)}}{Sales_{(t)}}}$$

GMI greater than 1 means gross margin is deteriorating

Asset Quality Index (AQI)

$$AQI = \frac{1 - \frac{(Current\ Assest_{(t)} + netPPE_{(t)})}{Total\ Assests_{(t)}}}{1 - \frac{(Current\ Assest_{(t-1)} + netPPE_{(t-1)})}{Total\ Assests_{(t-1)}}}$$

AQI greater than 1 may indicate the tendencies of capitalizing and deferring costs that should have been expensed

Sales Growth Index (SGI)

$$SGI = \frac{Sales_{(t)}}{Sales_{(t-1)}}$$

SGI greater than or less than 1 may indicate that the firm is under possible pressure to manipulate earnings to keep up appearances

Financial Ratios

Depreciation Index (DEPI)

$$DEPI = \frac{\frac{Depreciation\ Expense_{(t-1)}}{(Depreciation\ Expense_{(t-1)} + netPPE_{(t-1)})}}{\frac{Depreciation\ Expense_{(t)}}{(Depreciation\ Expense_{(t)} + netPPE_{(t)})}}$$

DEPI greater than 1 may indicate tendencies of the assets being depreciated at a slower rate to boost earnings

Sales and General Administrative (SGAI)

$$SGAI = \frac{\frac{SGAIExpense_{(t)}}{Sales_{(t)}}}{\frac{SGAIExpense_{(t-1)}}{Sales_{(t-1)}}}$$

SGAI less than 1 may indicate that the company may manipulate earnings to defer costs

Accruals to Asset Ratio (ACCR)

$$ACCR = \frac{Profit\ after\ Tax_{(t)} - Cash\ from\ Operations_{(t)}}{Total\ Assests_{(t)}}$$

ACCR greater than 1 may indicate that the accruals can possibly be used to manipulate earnings

Leverage Index (LEVI)

$$LEVI = \frac{\frac{(LTD_{(t)} + CurrentLiabilities_{(t)})}{Total\ Assests_{(t)}}}{\frac{(LTD_{(t-1)} + CurrentLiabilities_{(t-1)})}{Total\ Assests_{(t-1)}}}$$

Data Collection

Data Collection

- Data around eight financial ratios collected.
- SEBI and Prowess database used as the source to collect data about public listed company in India.
- Out of 1239 observations, 39 observations which were identified as doing fraud by SEBI in earlier audits

Train Set

- 70% of the data used for model training (840 Non manipulators + 28 Manipulators = 868 observations)
- 5 cross validation set created for the 868 observations

Test Set

- 30% of the data used for model testing (360 Non manipulators + 11 Manipulators = 371 observations)

Applying Supervised Learning Algorithm to detect fraud

Data challenges and remedy

Model Accuracy and ROC curve may not be reliable measure to evaluate model performance

Data Bias

1239 observations

1200 non manipulators

39 manipulators

Sampling Strategy

Bootstrap with up sample

Bootstrap with down sample

Bootstrap with synthetic sample

Bootstrap with simulation based sample

Outcome of traditional models

Low specificity reported by logistic regression and Neural network

Model Performance on Company Financial Ratios

Model (70% Train, 30% Test)	Performance on 5 CV Data (28 – Yes, 840 – No)			Performance on Test Set (11- Yes, 360 – No)			ROC on Test Set
	Sen	Spec	Overall	Sen	Spec	Overall	
Logistic Regression	0.99	0.03	0.97	1.00	0.18	0.97	0.94
Neural Network	0.99	0.19	0.97	0.99	0.36	0.97	0.88
Sensitivity = Non-Manipulator (No), Specificity = Manipulator(Yes)							

Overall accuracy and ROC curve suggests a good performance of the model

Ensemble methods with sampling–Models applied

भारतीय प्रबंध संस्थान बेंगलूर
INDIAN INSTITUTE OF MANAGEMENT BANGALORE

Random Forest with bootstrap	Random Forest with up sample	Random Forest with down sample
Random Forest with synthetic sample	Random Forest with simulated sample	Ada boost with bootstrap
Ada boost with up sample	Ada boost with down sample	Ada boost with synthetic sample
Ada boost with simulated sample	XG boost with boot strap sample	XG boost with up sample
XG boost with down sample	XG boost with synthetic sample	XG boost with simulated sample

Performance measure

Sensitivity Specificity, Model accuracy

Bootstrap for random forest and simulation based sample

5 cross validation dataset for all other models

Test dataset:

Sensitivity, specificity, ROC curve

Performance measures from ensemble methods

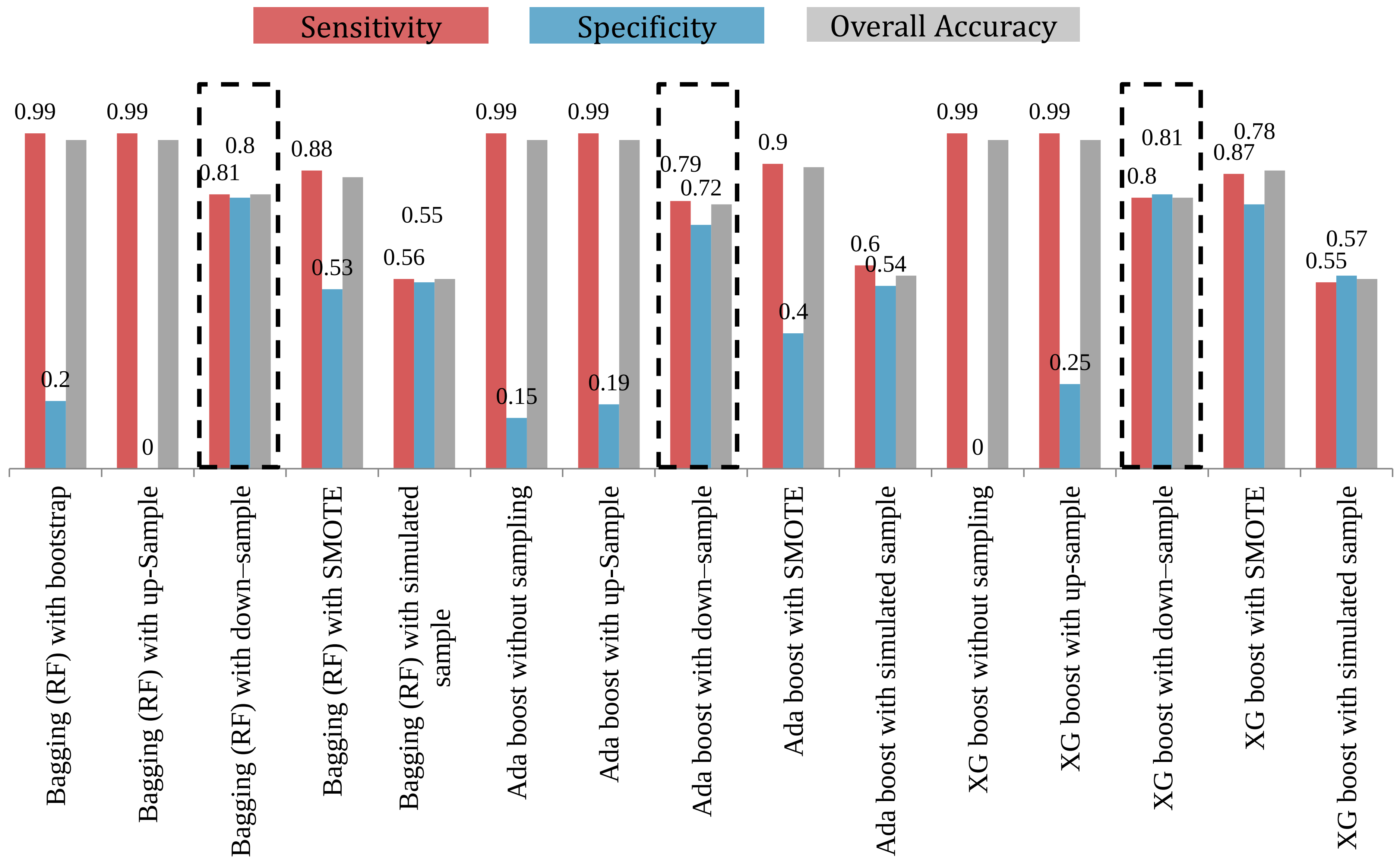
Model Performance on Company Financial Ratios

Model (70% Train, 30% Test)	Performance on 5 CV Data (28 – Yes, 840 – No)			Performance on Test Set (11- Yes, 360 – No)			ROC on Test Set
	Sen	Spec	Overall	Sen	Spec	Overall	
Bagging (RF) with bootstrap	0.99	0.20	0.97	0.99	0.18	0.97	0.95
Bagging (RF) with up-Sample	0.99	0.00	0.97	1.0	0.09	0.97	0.93
Bagging (RF) with down–sample	0.81	0.80	0.81	0.80	1.0	0.81	0.94
Bagging (RF) with SMOTE	**0.88**	**0.53**	**0.86**	**0.89**	**.082**	**0.89**	**0.93**
Bagging (RF) with simulated sample	**0.56**	**0.55**	**0.56**	**0.60**	**0.64**	**0.60**	**0.65**
Ada boost without sampling	**0.99**	**0.15**	**0.97**	**0.99**	**0.27**	**0.98**	**0.93**
Ada boost with up-Sample	**0.99**	**0.19**	**0.97**	**0.99**	**0.45**	**0.98**	**0.92**
Ada boost with down–sample	0.79	0.72	0.78	0.77	0.82	0.77	0.90
Ada boost with SMOTE	**0.90**	**0.40**	**0.89**	**0.91**	**0.73**	**0.91**	**0.91**
Ada boost with simulated sample	**0.60**	**0.54**	**0.57**	**0.60**	**0.82**	**0.61**	**0.74**
XG boost without sampling	**0.99**	**0.00**	**0.97**	**0.99**	**0.27**	**0.97**	**0.95**
XG boost with up-sample	0.99	0.25	0.97	0.99	0.46	0.97	0.96
XG boost with down–sample	0.80	0.81	0.80	0.75	0.91	0.76	0.96
XG boost with SMOTE	**0.87**	**0.78**	**0.88**	**0.86**	**0.73**	**0.85**	**0.91**
XG boost with simulated sample	**0.55**	**0.57**	**0.56**	**0.53**	**0.82**	**0.54**	**0.72**

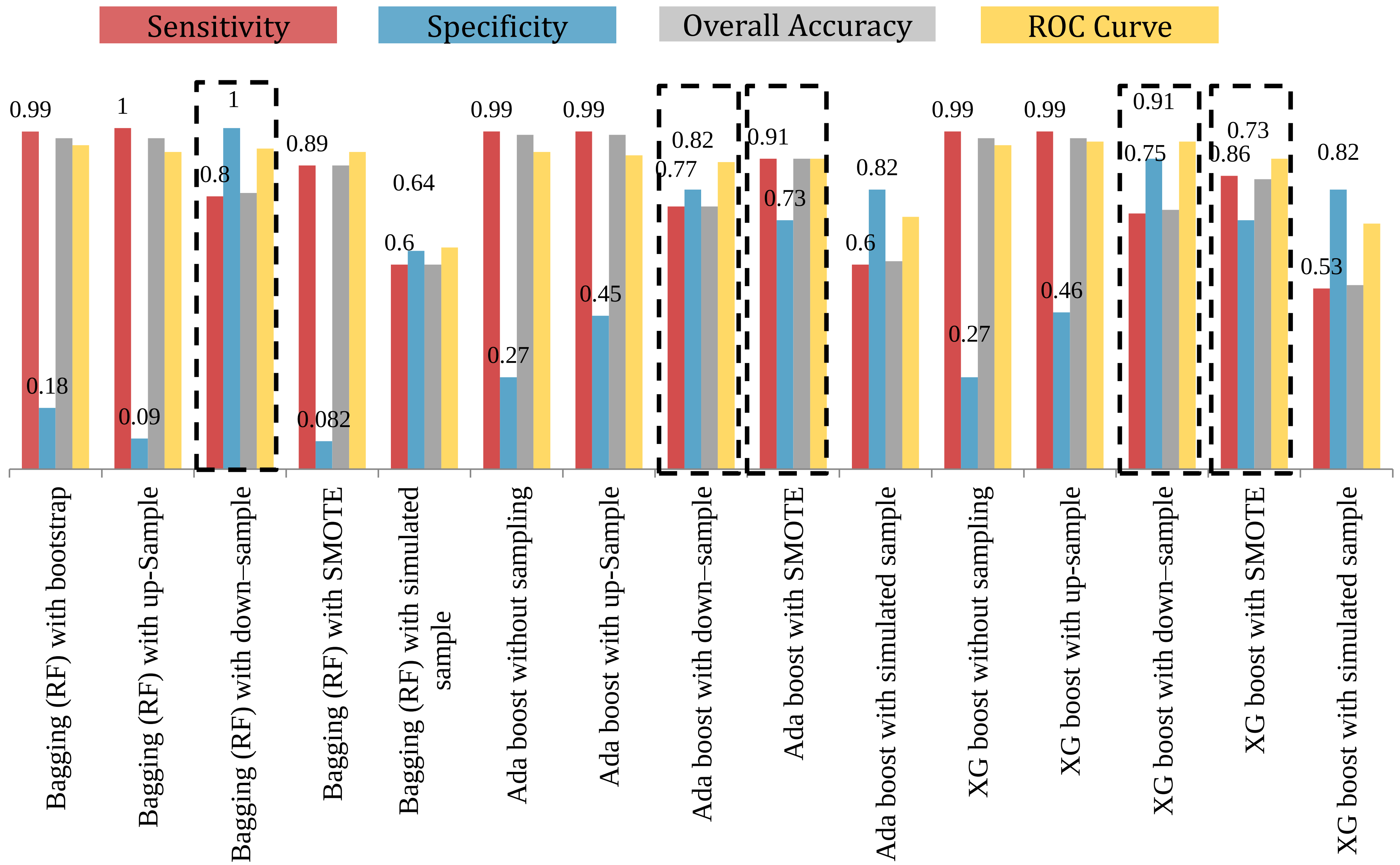
Sensitivity = Non-Manipulator (No), Specificity = Manipulator(Yes)

Performance on 5 CV set–Graphical View

भारतीय प्रबंध संस्थान बेंगलूर
INDIAN INSTITUTE OF MANAGEMENT BANGALORE



Performance on Test set–Graphical View



Dealing with Class Imbalance – Up-sample

- Up-sample strategy is applied on the data in such a way that the number of cases, that is, class frequencies for all the labels which do not belong to majority class, becomes equal to the majority class.
    - One of the ways to achieve this is by performing random sampling with replacement for all the cases which do not belong to the majority class.

- Up-sampling strategy balances out the class distribution, but since this strategy just duplicates the data (by repetition), it does not provide additional information to improve model performance.
    - Note that the percentage by which to up-sample the other classes need not necessarily be of the magnitude which makes the number of cases equal for all the classes.

Dealing with Class Imbalance – Down sample

- Down-sample strategy is applied on the data in such a way that the number of cases, that is, class frequencies for all the labels which do not belong to minority class becomes equal to the minority class.
    - One of the ways to achieve this is by performing random sampling with replacement or without replacement for all the cases which do not belong to the minority class.

- It reduces the overall number of cases in the data, it removes class imbalance issue and creates a heterogeneous mix from which algorithms can learn useful pattern..
    - Note that the percentage by which to down-sample the other classes need not necessarily be of the magnitude which makes the number of cases equal for all classes.

Dealing with Class Imbalance – SMOTE

Synthetic Minority Over-sampling Technique

- SMOTE() method from DMwR package in R implements synthetic sampling of minority class by setting the parameter perc.over. SMOTE() method also support under sample the majority class by setting the parameter perc.under.
  - perc.over: This parameter by default is set to 200. For each case in the original dataset belonging to the minority class, 200/100, that is, two new examples for each case will be created
  - perc.under: This proportion is calculated with respect to the number of newly generated minority class records. If 200 new records of minority class were generated, a value of perc.under of 200 will randomly select 200 cases belonging to the majority classes.

Dealing with Class Imbalance – SMOTE

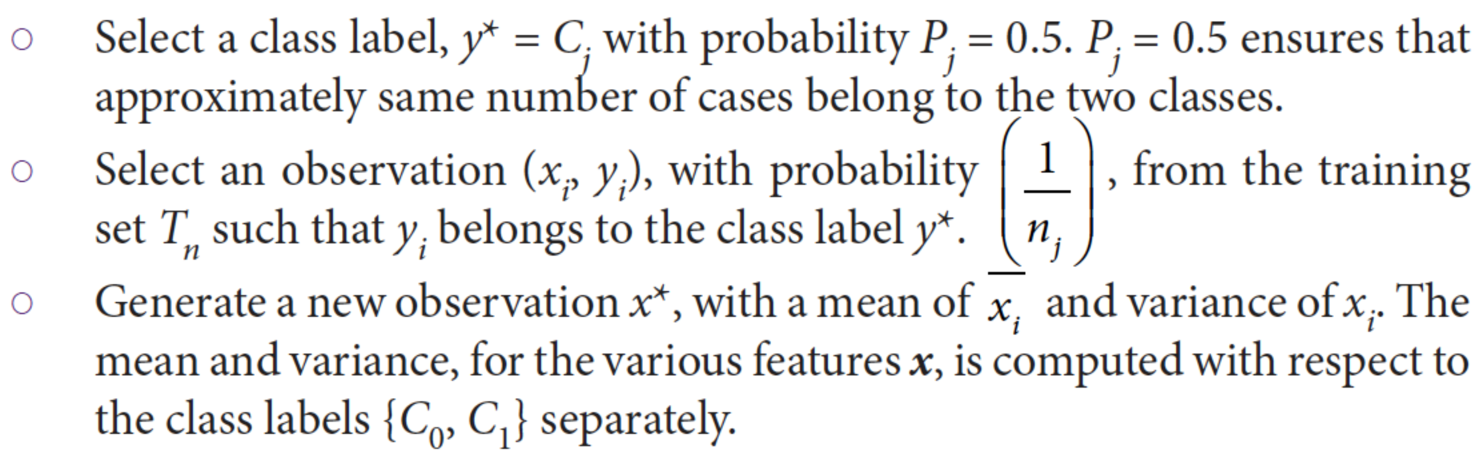
SMOTE synthesizes new examples for each minority case as follows:

- In the minority class, find k nearest neighbour to a given minority case (say A).
- Draw a line segment which joins any/all of the k minority class nearest neighbours to a given minority case (A).
- Depending on the number of cases to be over-sampled, select, at random, any/all of the k neighbours. Say, if amount of over-sampling needed is 300%, then three nearest neighbours from amongst the five neighbours to minority case (A), will be chosen and three random samples will be generated along the three line segments. The samples will be generated as follows:
    - Take the difference between the feature vector (e.g., case A) and one of the randomly selected nearest neighbours' feature vector (say B).
    - Multiply the difference by a random number between 0 and 1 (Xuet al. 2017). This will result in a new feature vector (say C).
    - Add the new vector C to the feature vector (e.g., case A). This will result in selection of a random case along the line segment joining A and B.
    - Repeat the above steps along the other two line segments to generate two more samples.

Dealing with Class Imbalance – ROSE

Random Over-Sampling Examples (ROSE): Let Tn be a training dataset with n observations:

- Let (**xi**, yi) denote an observation, where **x** denotes feature vector and Yi belongs to {C0, C1}, that is, binary class. Let the number of cases in class Cj, j = 0,1 be denoted by nj < n. The new sample is generated as follows:



- ○ Select a class label, $y^* = C_j$ with probability $P_j = 0.5$. $P_j = 0.5$ ensures that approximately same number of cases belong to the two classes.
- ○ Select an observation (x_i, y_i), with probability $\left(\frac{1}{n_j}\right)$, from the training set T_n such that y_i belongs to the class label y^*.
- ○ Generate a new observation x^*, with a mean of $\overline{x_i}$ and variance of x_i. The mean and variance, for the various features $\boldsymbol{x}$, is computed with respect to the class labels $\{C_0, C_1\}$ separately.

- Essentially, we pick an observation from the training set and generate a new sample (x*,y*) in its neighborhood. The shape of the neighbourhood is determined by the mean of the feature and the width is determined by the variance of the feature. ROSE uses simulated bootstrap approach to generate the new data points

For more details, refer Machine learning using R – Kumar Rahul and U Dinesh Kumar

Boosting: Adaboost

X1	X2	…	Ya	W1	Yp1	E1	MP	Non – normalized Weight	Normalized Weight (W2)	Bins
24	M		1	0.17	-1	0.17	0.35	0.23570	0.25	0 - 0.25
56	F		1	0.17	1	0.00	0.35	0.11785	0.125	0.25 - 0.375
34	M		-1	0.17	-1	0.00	0.35	0.11785	0.125	0.375 - 0.5
29	M		1	0.17	1	0.00	0.35	0.11785	0.125	0.5 - 0.625
45	F		-1	0.17	1	0.17	0.35	0.23570	0.25	0.625 - 0.875
37	M		-1	0.17	-1	0.00	0.35	0.11785	0.125	0.875 - 1
						0.33		0.94	1.00	

The classifier 1 steps are:

- Weight (W) = 1/n
- Total Error (TE) = sum(W(i) * Te(i)) / sum(W); Te(i) =1 if misclassified
    - Iteration 1: Total Error is 1/3
- Model Performance (MP or alpha) = ½*ln[(1-TE)/ TE]; MP can be –ve or +ve.
    - Iteration 1: MP = 0.35
- W= W* exp(-MP *ya*Yp)/NZ; where NZ is normalized weight
    - For observation 1: Non normalized weight = (1/6) * exp(-0.35*(+1)*(-1)
- Generate 6 random numbers: 0.4760, 0.0790, 0.8732, 0.3936, 0.9093, 0.581903

Boosting: Adaboost

X1	X2	…	Ya	W1	Yp1	E1	MP	Non – normalized Weight	Normalized Weight (W2)	Bins
34	M		-1	0.17	-1	0.00	0.79	0.07543	0.101190476	0-0.10
24	M		1	0.17	1	0.00	0.79	0.07543	0.101190476	0.10-0.20
45	F		-1	0.17	1	0.17	0.79	0.36827	0.494047619	0.20-0.69
34	M		-1	0.17	-1	0.00	0.79	0.07543	0.101190476	0.69-0.79
37	M		-1	0.17	-1	0.00	0.79	0.07543	0.101190476	0.79-0.89
29	M		1	0.17	1	0.00	0.79	0.07543	0.101190476	0.89-0.99
						0.17		0.75	1.00	

- With the above data, the steps explained in previous slides are repeated. This builds classifier 2
- The number of classifiers to be built can be a hyper-parameter which minimizes the exponential loss function

$$Objective\ function = \left(\frac{1}{m}\right) * \left(\sum_{i=1}^{m} e^{-y_a * C}\right)$$

Where, m is the number of observations and C is the final set of classifier:

$$C = sign\left(\sum_{i=1}^{C} MPc * Oc\right) where\ O\ is\ the\ output\ from\ Classifer\ i$$

Applying Unsupervised Learning Algorithm to detect fraud

Multivariate Gaussian Model

- Identify the features
    - 1239 observations across eight financial ratios
    - 1200 ($Y = 0;\ non\ manipulators$) and 39 ($Y = 1;\ manipulators$)
    - $training\ set = \{x^{(1)}, x^{(2)}, \dots.\, x^{(m)}\}\ where\ m = 900$ obs. across eight ratios
    - $test\ set = \{(x^1{}_{test}, y^1{}_{test}), (x^2{}_{test}, y^2{}_{test}), \dots.\,(x^m{}_{test}, y^m{}_{test})\}\ where\ m =$ 339 obs. across eight ratios (300 non manipulators and 39 manipulators)
- Compute mean and standard deviation for all the features in train set
- Given a test set compute

$$P(x) = \prod_{j=1}^{n} (1/\sqrt{2\pi}\,\sigma_j) * \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j{}^2}\right)\ where\ n\ is\ the\ number\ of\ feature$$

$$Y = \begin{cases} 1 & if\ P(x) < \in\ then\ anomaly \\ 0 & If\ P(x) \geq \in\ then\ ok \end{cases}$$

- Fine tune ∈ to catch anomaly

Outcome of Multivariate Gaussian models

Classification Matrix at ∈ 0.25

Actual Count		Count of Predicted Status		
		No	**Yes**	**Total**
	No	214	86	300
	Yes	10	29	39
	Grand Total	224	115	339

Model Performance on Company Financial Ratios

Model (70% Train, 30% Test)	Performance on Test Set (11- Yes, 339 – No)			ROC on Test Set
	Sen	**Spec**	**Overall**	
Gaussian Model	0.71	0.74	0.72	

Sensitivity = Non-Manipulator (No), Specificity = Manipulator(Yes)

Conclusion

Fraud is not a common phenomena and very few cases may come up in light.

Relying on metrics like ROC, model accuracy may be misleading.

Data preparation and pre-processing to remove the bias in data may be needed to give good learning experience to the machine.

Both supervised and unsupervised learning models may help identify fraud.

Thank you