SIGRNN: Synthetic Minority Instances Generation in Imbalanced Datasets using a Recurrent Neural Network

Reda Al-Bahrani[a], Dipendra Jha[b], Qiao Kang, Sunwoo Lee[c], Zijiang Yang, Wei-Keng Liao, Ankit Agrawal[d] and Alok Choudhary

Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, U.S.A.

Keywords: Synthetic Data, Balancing, Oversampling, Classification, Imbalanced Dataset.

Abstract: Machine learning models trained on imbalanced datasets tend to produce sub-optimal results. This happens because the learning of the minority classes is dominated by the learning of the majority class. Recommendations to overcome this obstacle include oversampling the minority class by synthesizing new instances and using different performance measures. We propose a novel approach to handle the imbalance in datasets by using a sequence-to-sequence recurrent neural network to synthesize minority class instances. The generative neural network is trained on the minority class instances to learn its data distribution; the generative neural network is then used to synthesize minority class instances; these instances are used to augment the original dataset and balance the minority class. We evaluate our proposed approach against several imbalanced datasets. We train Decision Tree models on the original and augmented datasets and compare their results against the Synthetic Minority Over-sampling TEchnique (SMOTE), Adaptive Synthetic sampling (ADASYN) and Synthetic Minority Over-sampling TEchnique-Nominal Continuous (SMOTE-NC). All results are an average of multiple runs and the results are compared across four different performance metrics. SIGRNN performs well compared to SMOTE and ADASYN, specifically in lower percentage increments to the minority class. Also, SIGRNN outperforms SMOTE-NC on datasets having nominal features.

1 INTRODUCTION

Classification datasets for training machine learning models are generally assumed to be balanced. A balanced dataset is composed of approximately an equal number of instances from each class. However, some scientific and real-world datasets are highly imbalanced. The ratio between some classes in these datasets can be quite high. Machine learning models trained on such imbalanced datasets tend to produce sub-optimal results with inappropriate prediction accuracy (Visa and Ralescu, 2005; Maloof, 2003). Since the models focus on learning the data representation of the majority class, they tend to neglect the data representation of the minority classes (Japkowicz et al., 2000; Japkowicz and Stephen, 2002). There exist several research works that have investigated the problem of imbalanced datasets with machine learning algorithms such as neural networks and support vector machines (Fawcett and Provost, 1997; Chan and Stolfo, 1998; Kubat et al., 1997b).

Some existing approaches to overcome the challenge of imbalance in training datasets include re-sampling using unsupervised learning, undersampling the majority class, oversampling the minority class, synthesizing from the minority class, and using different performance measures (Yap et al., 2014; Nickerson et al., 2001; Drummond et al., 2003; Estabrooks et al., 2004). These approaches are based on either decreasing the number of instances in the majority class or increasing the number of instances the minority class. Usually, the minority class instances are incremented by either repeating the original instances or constructing new instances using nearest neighbor approach based on random subsets of instances.

Synthetic Minority Oversampling TEchnique (SMOTE) increases the minority class by creating synthetic instances based on the k-nearest neighbor instances in the minority class (Chan and Stolfo, 1998). SMOTE, as demonstrated on multiple datasets

[a] https://orcid.org/0000-0002-1528-0792
[b] https://orcid.org/0000-0002-6210-1937
[c] https://orcid.org/0000-0001-6334-3068
[d] https://orcid.org/0000-0002-5519-0302

Al-Bahrani, R., Jha, D., Kang, Q., Lee, S., Yang, Z., Liao, W., Agrawal, A. and Choudhary, A.
SIGRNN: Synthetic Minority Instances Generation in Imbalanced Datasets using a Recurrent Neural Network.
DOI: 10.5220/0010348103490356
In *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2021)*, pages 349-356
ISBN: 978-989-758-486-2

by the authors, can handle continuous variables only. Synthetic Minority Over-sampling TEchnique-Nominal Continuous (SMOTE-NC) is a variant presented in the paper able to handle nominal data. SMOTE-NC was tested by the authors on a single dataset, the Adult dataset. SMOTE and SMOTE-NC cannot generate synthetic instances in a dataset containing only nominal features. To our knowledge, SMOTE and many SMOTE variants (Han et al., 2005; Ramentol et al., 2012; Maciejewski and Stefanowski, 2011) can only operate on datasets containing numerical features. Adaptive synthetic sampling (ADASYN) (He et al., 2008) is another approach where the algorithm tries to learn examples that are harder to generate in the minority class.

In this paper, we present a novel approach of **S**ynthetic minority **I**nstances **G**eneration using **R**ecurrent **N**eural **N**etwork (SIGRNN) to handle the imbalance in datasets. The proposed approach utilizes an encoder-decoder recurrent neural network to generate synthetic instances from the minority class population. Instances in the minority class of the dataset are treated as a fixed length set of features where each feature is represented by a set of tokens. By treating each instance in the minority class as a small set vocabulary (a sentence), the sequence-to-sequence encoder-decoder recurrent neural network is trained to predict the next token based on the current and past input tokens of a sentence. The SIGRNN model is trained only on the minority class to augment the training datasets by generating synthetic minority instances We evaluate our approach using three different datasets. These datasets represent different feature types, minority class to majority class ratios, and minority class sizes.

To analyze the impact of the proposed data augmentation approach, Decision Tree models were trained on the original dataset and the augmented datasets, and the results were compared against SMOTE, ADASYN, and SMOTE-NC depending on the input dataset. Metrics such as Accuracy, Area under the ROC Curve, F1 score, and Gmean were compared.

2 RELATED WORK

2.1 Over-sampling

Synthetic Minority Over-sampling TEchnique (SMOTE) presented in (Chawla et al., 2002) over-samples the minority class by creating "synthetic" examples. SMOTE operates on the features by taking each minority class instance and introducing synthetic instances along the line segments joining any/all of the k minority class nearest neighbors. A number of synthetic instances are generated based on the k nearest neighbors of each instance in the minority class. In case there exist nominal features in the data, Synthetic Minority Over-sampling TEchnique-Nominal Continuous (SMOTE-NC) populates nominal features by selecting the value occurring in the majority of the k-nearest neighbors. Adaptive Synthetic Sampling approach (ADASYN) is another approach where the algorithm focuses on learning examples that are hard to generate in the minority class while focusing less on generating examples that are easy to learn.

2.2 Performance Measures

Several performance measures have been proposed to measure the effectiveness of machine learning classifiers on the minority class. The area under the receiver operating characteristic curve is by far the most used. The AUC ROC represents the relationship between sensitivity and specificity (Beck and Shultz, 1986). The F1-score captures the relationship between precision and recall. Other measures have been proposed such as the geometric mean (Kubat et al., 1997b).

2.3 Language Modeling

In statistical language modeling recurrent neural networks have been used to learn a representation of words by training on a large corpus (Bengio et al., 2003). Such models are capable of learning the probability of word sequences. In (Cho et al., 2014) the authors propose a recurrent neural network encoder-decoder architecture capable of learning mappings of input sequences to an output sequence. The concept of sequence-to-sequence recurrent neural networks since has been used for tasks such as language translation (Sutskever et al., 2014) and generating image descriptions (Karpathy and Fei-Fei, 2015).

3 SYNTHETIC MINORITY INSTANCES GENERATION USING A RECURRENT NEURAL NETWORK

The overall process of the SIGRNN approach is composed of four main steps - feature ordering, building the corpus and input data for training the recurrent neural network, model selection and training, and data instance generation using the trained model. These steps are explained below.

3.1 Entropy based Feature Ordering

Text inputs in general RNN training follow the syntax and semantics of the language. The same word can be used at different positions in the input with/without altering the meaning of the sentence; the RNN model would automatically capture the semantics of the language from the input sentences. In our case, the input is not from a natural language; rather the input sentences are records, the feature order is based on the original dataset. The order of features from the original dataset does not have any particular meaning, they can be ordered in any way and used as input to train our model. However, the order of features in the input can be critical to the learning of the RNN model. Some features can be easy to learn, only having some specific range/list of values for our minority class, while the model can get confused with other features having a broader range of values. To handle this, we used an entropy-based feature ordering for our input training data.

We apply an entropy ordering method to generate new records. A parameter setting for the LSTM model can encode the joint probability distribution of all features. Having an ordered set of features $\{X_1,..X_n\}$, an LSTM model can encode the distributions of $P_c(X_i|X_1,..,X_{i-1})\forall i \in [1,n]$ for a particular class c. The joint distribution of all features can be derived as the following.

$$P_c(X_1 = x_1,..,X_n = x_n) = \prod_{i=1}^{n} P_c(X_i = x_i|X_1 = x_1,..,X_{i-1} = x_{i-1}) \quad (1)$$

Based on $P_c(X_1,..,X_n)$, denoted as P, we can generate random sequences of features for future training. It is obvious that the computation of feature joint distribution depends on the ordering of features. If the LSTM models can perfectly model the conditional distributions, the ordering does not matter. However, this assumption is usually unrealistic, especially when the instance is imbalanced. Let ε be an upper bound error associated with the conditional probability, assuming $\varepsilon + p_i < 1$ and $p_i - \varepsilon > 0\ \forall 1 \leq i \leq n$. We want to minimize the error term in the final joint probability distribution, denoted as $\Delta(X_1,...,X_n)$. We use notation $p_i = P_c(X_i|X_1,...,X_{i-1})$ for simplicity.

$$\Delta(X_1,...,X_n) = |\prod_{i=1}^{n}(p_i \pm \varepsilon) - P| \quad (2)$$

However, p_i and p_j are not independent of each other, since they both depend on the choice of feature ordering. As a result, we cannot simply minimize each of the p_i terms. An exhaustive search for the feature ordering has a factorial complexity with respect to the number of features, which is not feasible in reality.

To solve this problem, we apply a greedy heuristic search approach based on entropy. The entropy of p_i, denoted as $H(p_i)$, measures the degree of randomness of a probability distribution. It has the following property: If the probabilities of k random variables are all close to $\frac{1}{k}$, the entropy approaches to the maximum entropy $\log(k)$. Moreover, $\Delta(x_1,...,x_n)$ is minimized if all terms p_i are as close to uniform distribution as possible. Thus, maximizing $\sum_{i=1}^{n} H(p_i)$ is equivalent to minimizing $\Delta(X_1,...,X_n)$. Hence we order the features based on the conditional entropy p_i using a Decision Tree method. This approach is equivalent to a greedy search of n layers using entropy of $p_i \forall i \in [1,n]$ as heuristics.

We trained a Decision Tree based on entropy as the decision criterion on our original training data. In a Decision Tree, the feature entropy decreases from root to leaves; the features at the top have high entropy while the leaves have an entropy of zero. We use the entropy computed by a Decision Tree to sort our input features from highest to lowest entropy. The model is provided with features using this order during training.

3.2 Building the Corpus and Input Data

We convert the minority class instances into a corpus to train the SIGRNN. An instance $\mathcal{R}$ consists of $\mathcal{X}$ features: categorical and numerical features. Each value $\mathcal{V}$ in a feature is considered to be a word in the vocabulary. To uniquely identify a feature based on its value, the words are assigned tokens. For instance, one of our features is GENDER, having two categories- 1 for MALE and 2 for FEMALE, two tokens will be generated for the GENDER, these tokens are: GENDER_MALE and GENDER_FEMALE. After these tokens are generated we use them to convert instances in the feature space to sentences in the language space.

The final corpus is composed of sentences containing $\mathcal{X}$ words (tokens) where each word represents a feature value and the sentence represents an instance. The corpus generated consists of sentences corresponding to the minority class instances in the training dataset. Next, we build a vocabulary to uniquely map each word in our training dataset into embeddings and convert our input sentences into word vectors. These word vectors are fed into the SIGRNN for training.

3.3 Model Training and Instance Generation

The SIGRNN is composed of an encoder-decoder model using Long-short Term Memory (LSTM) cells to predict the next feature based on the observed features. The encoder is basically an embedding look-up table for the input word vectors; it converts the inputs from high dimension due to large size of vocabulary to a reduced representation. A sequence to sequence decoder is used on top of the encoder output, and is composed of multiple LSTM layers. We experimented with different number of LSTM cells in different layers and different hyper-parameters to fine tune our model; the experiments presented here used a decoder layer, two layers of 512 or 1,024 LSTM cells, and a decoder, with a mini batch size of 8 or 32 depending of the dataset size. A fully connected layer with softmax activation is used on the decoder output to get the probability of output words. The model architecture is shown in Figure 1. The embedding size used was the same size of the number of tokens in the minority class of the training set. The models were trained for 15 epochs. Once the models were trained, sequences of tokens are then generated using the network. Finally, the generated tokens are converted back from the language space to the feature space and the training datasets were augmented from the generated instances.

3.4 Evaluation Approach

We trained Decision Tree models to evaluate and compare the impact of data augmentation using the SIGRNN against SMOTE, ADASYN, and SMOTE-NC. The models are trained and selected through a 10-fold cross-validation; each experiment is run with 10 different random seeds. The results are an average over the 10 runs. This comparison is carried out to evaluate the data generation and not the machine learning algorithm itself. We use Decision Trees as the baseline model and use it throughout evaluation of datesets generated by SMOTE, ADASYN, and SMOTE-NC, and our proposed recurrent neural network approach.

The network is coded using PyTorch (Paszke et al., 2017) and is trained using an Nvidia GPU (GTX TitanX). To build the Decesion Tree models h2o platform (Candel et al., 2016) has been used, and to balance using SMOTE, ADASYN, and SMOTE-NC we used the implementation of (Lemaître et al., 2017).

4 RESULTS

In this section, we evaluate the efficiency of our proposed approach by analyzing two factors. First, we compare the quality of the generated data compared to the original minority class data. Second, we evaluate SIGRNN against SMOTE, and ADASYN by comparing Decision Tree performance across different performance metrics. Also, we compare SIGRNN and SMOTE-NC on datasets that contain nominal features.

4.1 Performance Metrics

Decision Trees were used as the baseline model in all experiments. The built models were evaluated on the following performance metrics:

1. **Accuracy:** It is the fraction of correctly classified examples in the test set.

$$Accuracy = \frac{correct predictions}{total number of predictions} \tag{3}$$

2. **Area under the Receiver Operating Characteristic Curve:** The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

3. **F1-score**: It is a measure of a test's accuracy. It considers both precision and recall of the test to compute the score. The F1 score is the harmonic average of the precision and recall.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

4. **Geometric Mean of Class Accuracy:** It combines the positive class accuracy (PA) and the negative class accuracy (NA).(Kubat et al., 1997a)

$$Gmean = \sqrt{PA \times NA} \tag{5}$$

4.2 Datasets

We present results for three different datasets, these datasets are publicly available (Dua and Graff, 2017). Table 1 describes the datasets used in our experiments as below.

1. **SATIMAGE:** This database consists of *t* values of pixels in 3 by 3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The goal is to classify the pixel, given the multi-spectral values. To generate an imbalanced dataset all classes were collapsed except for class 4.

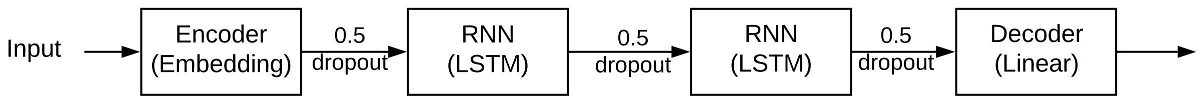


Figure 1: An overview of the recurrent neural network architecture used to generate synthetic minority class instances.

2. **HABERMAN:** This dataset contains breast cancer cases from the Billings Hospital at the University of Chicago. The two classes present in the dataset indicate survival of 5 years of patients who had undergone surgery for breast cancer between 1958 and 1970. The data consists of four attributes: age of patient at time of operation, year of operation, number of positive axillary nodes detected, and survival status.
3. **ADULT:** The Adult dataset consists of Census Income information and is used to predict if an individual's income is greater than $50K/yr. The information was extracted from the 1994 census bureau database. It contains information of working adults between the ages of 16 and 100. The dataset contains 48,842 records, each record containing five numerical and eight categorical features.

Table 1: Description of the datasets used in our experiments.

		Train	**Test**
Name	**Types**	**Maj/Min**	**Maj/Min**
Satimage	Num	4k/415	1.7k/211
Haberman	Num	179/65	46/16
Adult	Num/Cat	24.7k/7.8k	12.4k/3.8k

4.3 Quality of Generated Data

We compare the quality of the generated instances by comparing their distribution with the original minority class instances. We took two features from the ADULT dataset and generated histogram diagrams. The blue histograms are for SMOTE-NC, red for SIGRNN, and black is the original data. In Figure 2 the original minority class and the generated minority class instances are overlayed to compare the distribution of the two features. In both features SIGRNN generates data that spans the whole set of bins, while SMOTE-NC struggles to generate values in the under-represented bins i.e. the bins on the two tails of the distribution.

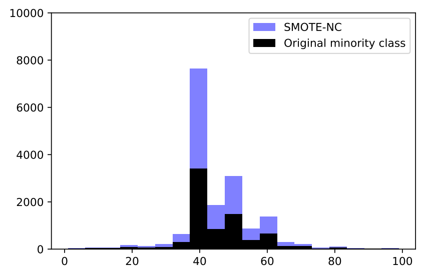
4.4 Performance Improvement

We compared the performance improvements by training Decision Tree models on the augmented dataset. For this, we train the Decision Tree models on training datasets created using different amount of synthetic instances (from 100% to 900% depending on the dataset size). We compare the performance of our models trained on data augmented using recurrent neural network to the same models trained on the data augmented using SMOTE, ADASYN, and SMOTE-NC.

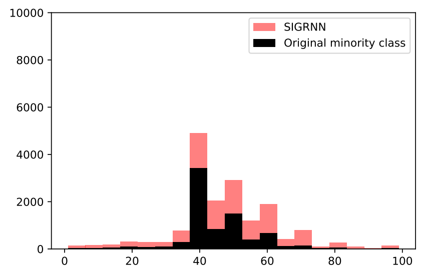
We performed a 10-fold cross validation with hyper-parameter tuning. The best model from 10-fold cross validation was selected. The final model is then used to generate performance metrics on the test set. This process was repeated at every increment in the minority class. First, Table 2 shows the comparison of SIGRNN and SMOTE-NC on the Adult dataset. SIGRNN performs well on datasets containing nominal data. We suspected that there will be a performance gap between SIGRNN and SMOTE-NC, since SMOTE-NC selects the value occurring the most in the k-nearest neighbors for nominal values while SIGRNN produces a value based on the sequence of prior features while generating the instance. Second, in Table 3 and 4 we compare the performace of the two datasets consisting of only continuous features over multiple performance metrics. Looking at the Gmean metric our proposed method outperforms SMOTE and ADASYN in most cases. Area under the ROC also shows improvement over other algorithms in most increments. In the case of Accuracy and F1 score, SIGRNN either matches or slightly trails SMOTE and ADASYN. SIGRNN performance can be compared to SMOTE even in cases where the training minority class is small.

5 CONCLUSION AND FUTURE WORK

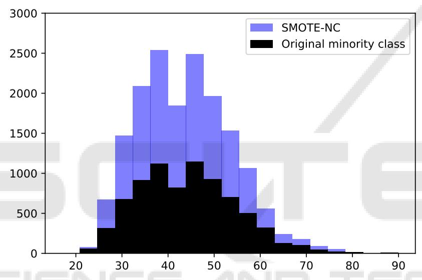
We formalize a method to handle imbalance in datasets utilizing a language model approach by converting a dataset to a corpus and then applying a sequence-to-sequence generative neural network to generate new sentences in the corpus. The generated corpus is then converted back to the original feature space. The transformation from feature space to corpus and back again to feature space produces promising results to tackle imbalanced datasets. We evaluated this method using multiple datasets of dif-



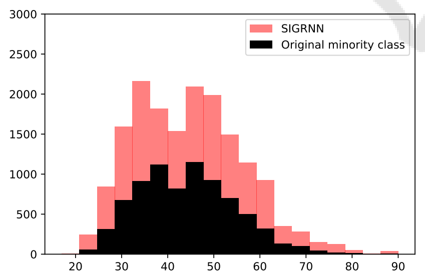
(a) ADULT histogram of hours per week



(b) ADULT histogram of hours per week



(c) ADULT histogram of age



(d) ADULT histogram of age

Figure 2: Histograms showing distribution of the actual minority class vs. the synthetic generated instances for two features from the ADULT dataset. The blue histograms are for SMOTE-NC, red for SIGRNN, and black is the original data. This overly is to compare the distribution of the two features. In both features SIGRNN generates data that spans the whole set of bins, while SMOTE-NC struggles to generate values in the under-represented bins.

ferent sizes, and features types. We demonstrated that the approach works well compared to SMOTE, ADASYN, and SMOTE-NC. Although, we handle numerical attributes in our proposed implementa-

Table 2: The adult dataset consists of a mixture of continuous and nominal features. Adult-1 is the original form of the dataset while Adult-2, and Adult-3 are created by removing continuous and nominal features respectively. Features were removed to demonstrate the behaviour of SMOTE-NC (S-NC) compared to SIGRNN in handling nominal features. Where * is present the variance is ± 0.01.

Adult-1					
%	**Method**	**Acc**	**AUC**	**F1**	**Gm**
0	-	0.83	0.81*	0.89	0.75*
100	SIGRNN	**0.81**	**0.82**	**0.87**	**0.78***
	S-NC	**0.81**	**0.82**	**0.87**	**0.78**
200	SIGRNN	0.8	**0.83**	0.86	**0.79**
	S-NC	**0.81**	0.82	**0.87**	**0.79**
300	SIGRNN	**0.8**	**0.83**	**0.86**	**0.79**
	S-NC	**0.8**	0.82	**0.86**	**0.79**
Adult-2					
%	**Method**	**Acc**	**AUC**	**F1**	**Gm**
0	-	0.8	0.77	0.87	0.68*
100	SIGRNN	**0.79**	**0.79***	**0.86**	0.71*
	S-NC	0.78	0.78	0.85	**0.72**
200	SIGRNN	**0.77**	**0.79**	**0.84**	0.72
	S-NC	0.75*	0.78	0.82*	**0.73**
300	SIGRNN	**0.77**	**0.8**	**0.84**	0.72
	S-NC	0.75*	0.78	0.82*	**0.74**
Adult-3					
%	**Method**	**Acc**	**AUC**	**F1**	**Gm**
0	-	0.82	0.8	0.88	0.74
100	SIGRNN	**0.81**	**0.82**	**0.87**	0.76
	S-NC	0.8	0.8	0.86	0.77
200	SIGRNN	**0.79**	**0.82**	**0.86**	**0.78**
	S-NC	**0.79**	0.81	0.85	**0.78**
300	SIGRNN	**0.79**	**0.82***	**0.85**	**0.78**
	S-NC	0.78	0.81	**0.85**	**0.78**

tion, this approach can be improved by adopting a branched recurrent neural network where each data type is handled by a branch to avoid converting numerical values to tokens before training the SIGRNN. We plan to experiment with adversarial training in future and train on both classes to generate better minority class instances.

ACKNOWLEDGMENT

This work is supported in part by the U.S. Department of Energy award numbers DE-SC0014330 and DE-SC0019358 and National Institute of Standards and Technology award number 70NANB19H005.

Table 3: Results for the decision tree on the Haberman dataset. Each result presents is the average of 10 different runs and the associated standard deviation. We can ovserve that SIGRNN performs well on the AUC ROC, and Gmean metrics in earlier increments. SIGRNN performs well as good as the other balancing methods on the F1 metric.

Haberman					
%	**Method**	**Acc**	**AUC**	**F1**	**Gmean**
0	-	0.69 ±0.05	0.61 ±0.05	0.79 ±0.04	0.57 ±0.06
100	SIGRNN	**0.65** ±0.03	**0.59** ±0.04	**0.76** ±0.03	**0.56** ±0.06
	SMOTE	**0.65** ±0.04	0.58 ±0.04	**0.76** ±0.03	0.55 ±0.05
	ADASYN	**0.65** ±0.06	0.58 ±0.07	0.75 ±0.05	0.55 ±0.08

Table 4: Results for the decision tree on the Satimage dataset. Each result presents is the average of 10 different runs and the associated standard deviation. We can ovserve that SIGRNN performs well on the AUC ROC, and Gmean metrics in earlier increments. SIGRNN performs well as good as the other balancing methods on the F1 metric.

Satimage					
%	**Method**	**Acc**	**AUC**	**F1**	**Gmean**
0	-	0.9±0.01	0.74±0.01	0.95±0.0	0.71±0.02
100	SIGRNN	**0.9**±0.01	**0.77**±0.02	**0.94**±0.0	**0.75**±0.03
	SMOTE	**0.9**±0.01	**0.77**±0.01	**0.94**±0.0	**0.75**±0.02
	ADASYN	**0.9**±0.01	0.76±0.02	**0.94**±0.0	0.74±0.03
200	SIGRNN	0.89±0.01	**0.78**±0.02	0.94±0.0	**0.76**±0.02
	SMOTE	**0.9**±0.01	0.77±0.02	0.94±0.0	**0.76**±0.02
	ADASYN	**0.9**±0.01	0.77±0.01	**0.95**±0.0	0.75±0.01
300	SIGRNN	**0.89**±0.01	**0.79**±0.01	**0.94**±0.0	**0.79**±0.02
	SMOTE	**0.89**±0.01	0.78±0.02	**0.94**±0.0	0.77±0.02
	ADASYN	**0.89**±0.01	0.77±0.02	**0.94**±0.0	0.76±0.03
400	SIGRNN	0.88±0.01	**0.79**±0.01	0.93±0.0	**0.79**±0.01
	SMOTE	**0.9**±0.01	**0.79**±0.01	**0.94**±0.0	0.78±0.02
	ADASYN	0.89±0.01	0.78±0.01	**0.94**±0.0	0.77±0.02
500	SIGRNN	0.88±0.01	**0.8**±0.02	0.93±0.0	**0.8**±0.02
	SMOTE	**0.89**±0.01	0.78±0.01	**0.94**±0.0	0.79±0.01
	ADASYN	0.88±0.0	**0.8**±0.01	0.93±0.0	0.79±0.01
600	SIGRNN	0.88±0.01	**0.8**±0.02	0.93±0.01	**0.8**±0.01
	SMOTE	**0.89**±0.01	0.79±0.01	**0.94**±0.0	**0.8**±0.02
	ADASYN	0.87±0.01	**0.8**±0.01	0.93±0.01	0.79±0.01
700	SIGRNN	**0.88**±0.01	0.8±0.01	**0.93**±0.0	**0.81**±0.01
	SMOTE	**0.88**±0.01	0.8±0.01	**0.93**±0.0	0.8±0.01
	ADASYN	**0.88**±0.01	**0.81**±0.02	**0.93**±0.0	0.8±0.02
800	SIGRNN	**0.88**±0.0	0.81±0.01	**0.93**±0.0	0.81±0.01
	SMOTE	**0.88**±0.01	0.8±0.01	**0.93**±0.0	0.8±0.01
	ADASYN	**0.88**±0.01	**0.82**±0.01	**0.93**±0.0	**0.83**±0.01
900	SIGRNN	0.87±0.01	0.8±0.02	**0.93**±0.0	0.81±0.02
	SMOTE	**0.88**±0.01	0.81±0.01	**0.93**±0.0	0.81±0.01
	ADASYN	**0.88**±0.01	**0.83**±0.01	**0.93**±0.0	**0.84**±0.01

REFERENCES

Beck, J. R. and Shultz, E. K. (1986). The use of relative operating characteristic (roc) curves in test performance evaluation. *Archives of pathology & laboratory medicine*, 110(1):13–20.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Candel, A., Parmar, V., LeDell, E., and Arora, A. (2016). H2o.ai. *H2O. ai Inc*.

Chan, P. K. and Stolfo, S. J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *KDD*, volume 98, pages 164–168.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority oversampling technique. *Journal of artificial intelligence* *research*, 16:321–357.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Drummond, C., Holte, R. C., et al. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.

Japkowicz, N. et al. (2000). Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA.

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Kubat, M., Holte, R., and Matwin, S. (1997a). Learning when negative examples abound. In *European Conference on Machine Learning*, pages 146–153. Springer.

Kubat, M., Matwin, S., et al. (1997b). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Nashville, USA.

Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Maciejewski, T. and Stefanowski, J. (2011). Local neighbourhood extension of smote for mining imbalanced data. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 104–111. IEEE.

Maloof, M. A. (2003). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, volume 2, pages 2–1.

Nickerson, A., Japkowicz, N., and Milios, E. E. (2001). Using unsupervised learning to guide resampling in imbalanced data sets. In *AISTATS*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. *xyz*.

Ramentol, E., Caballero, Y., Bello, R., and Herrera, F. (2012). Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and information systems*, 33(2):245–265.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Visa, S. and Ralescu, A. (2005). Issues in mining imbalanced data sets-a review paper. In *Proceedings of the sixteen midwest artificial intelligence and cognitive science conference*, volume 2005, pages 67–73. sn.

Yap, B. W., Rani, K. A., Rahman, H. A. A., Fong, S., Khairudin, Z., and Abdullah, N. N. (2014). An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In *Proceedings of the first international conference on advanced data and information engineering (DaEng-2013)*, pages 13–22. Springer.