Fraud Detection on Extremely Skewed Data

Diego Cervera de la Rosa, Eugenio Neira Bustamante, Gregory Walsh, and Guillermo Romero Moreno

***Abstract*—This work aims to analyse the performance of a set of popular machine learning algorithms on an extremely skewed machine learning problem; namely, fraud detection on a data-set of credit card transactions. The problem is remarkably difficult due to the scarcity of the positive class (around 500 samples), which makes training and testing challenging. We find that one-class support vector data descriptors showed the best performance when recall is highly prioritised, and XGBoost ensemble methods provided a modestly better performance otherwise, although not to a statistically significant degree, due to variance in results.**

I. Introduction

Fraud detection is an essential activity for most financial institutions. In a 2017 report, the Financial Fraud Action UK, a research organisation representing banks and payment processors, found that 80% of monetary losses due to fraud, equivalent to 618.0 million pounds, were a result of the misuse of payment cards [20]. Hence, technology which can automatically classify transactions as legitimate or fraudulent in real-time has the potential to cut losses significantly for institutions and customers.

As with other classification problems, given a sufficiently large training set of labelled examples, machine learning (ML) models can be trained to distinguish fraudulent transactions from legitimate ones. However, since the number of fraudulent transactions tends to be orders of magnitude smaller than those which are legitimate, inevitably fraud datasets are extremely imbalanced. Class imbalance within a dataset requires special treatment of the data, and a selection of an appropriate performance metric, in order to achieve a desirable balance of type I (false positive) and type II (false negative) errors.

In this paper, we aim to identify preprocessing, sampling, and modelling techniques suitable for dealing with a credit card fraud dataset containing highly imbalanced classes. Furthermore, we explain how, through the selection of an appropriate choice of performance metric, a financial institution could balance the cost associated with assessing potential fraudulent transactions against potential losses.

II. The dataset

For the purposes of this analysis, the "Credit Card Fraud Detection" dataset, available on Kaggle[1] was used [16]. It consists of the transactions made by European cardholders, occurred in a two day span, in September 2013. There are 284,807 transactions, of which only 492 were reported to be fraudulent. The positive class (frauds) represents just 0.172% of all transactions, making the dataset highly unbalanced.

To preserve the anonymity of cardholders, the publishers of the dataset applied principal component analysis to it before making it available. As a result, the published features include the first 28 principal components of this analysis, as shown in Figure 1. The dataset also contains two unmodified features, "time" (seconds elapsed since midnight) and "amount".

The high dimensionality of the dataset in conjunction with the small sample size of the fraud data, make the *curse of dimensionality* to be strongly present in the problem [8]. The curse of dimensionality implies the units of a grid space grow exponentially with the number of dimensions, and therefore the sparsity of the data distribution makes learning without overfitting a difficult task [11].

[1]Kaggle is a website which hosts datasets and competitions for data scientists (www.kaggle.com)

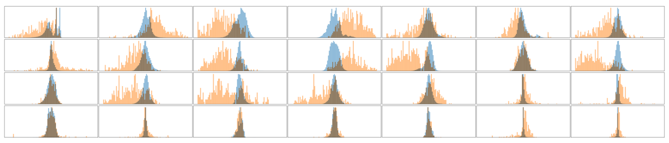

Fig. 1. Distribution of the features coming from Principal Component Analysis (blue: negative class, orange: positive class)

III. Experimental Design

In this section, the approach used to generate the performance metrics of different classification methods is described, along with the controls which were put in place to make

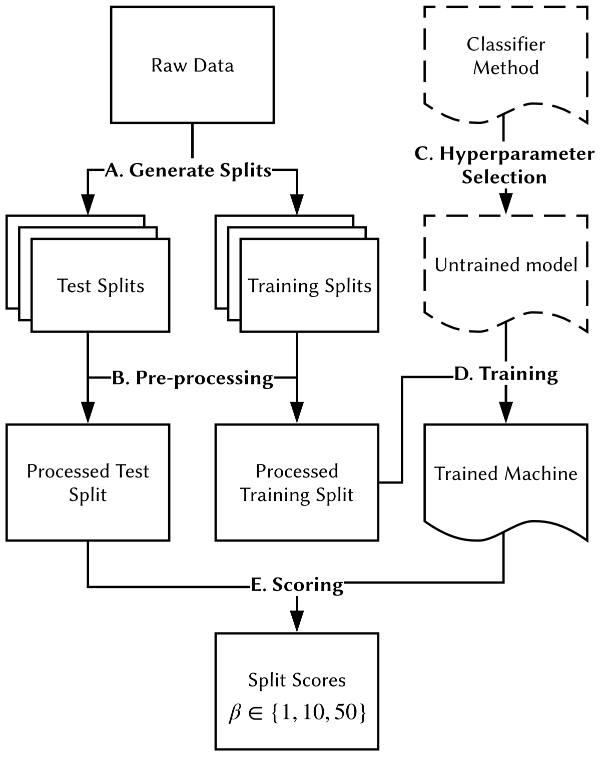

Fig. 2. Process diagram showing all stages of the evaluation process.

comparisons between methods as fair as possible. The indices of the subsections in this section broadly correspond to the labels of the stages shown in Figure 2.

A. Cross validation

To determine more reliably the performance of each method, a 10 split cross validation process, using Mont-Carlo sampling, was performed to get the average score and a measure of the consistency. Furthermore, to be as fair as possible, all methods were trained and validated using identical splits.

Because of the small number of positives in the dataset, the split was chosen to be 5%-95% in validation and training respectively. The proportion of positives (0.172%), as found in the entire dataset, was held constant for both training and validation splits. In each fold, the train set contains 270567 transactions of which 467 are fraudulent, and the validation set 14240, with 25 positives.

B. Pre-Processing

Since the majority of modelling techniques investigated tend to perform better, or train faster when inputs are normalised [3], all features were centred at zero and scaled by the inverse of their standard deviations. The calculations were executed on each training split individually, and the values were then used to transform the corresponding validation sets, thereby appropriately treating the validation sets as if they were unseen data.

A small amount of feature engineering was also performed. First, time, originally recorded in seconds since midnight, was converted to a two dimensional quantity by taking the cosine and sine of the angle between an hour hand and midnight on an analogue 24hr clock face. This transformation was applied since 23:59 is much closer to reality to 00:01, a fact which is not well represented when time is measured in seconds since midnight. Secondly, it was noted that the amounts were log-normally distributed, and so a logarithmic transformation was applied to improve the subsequent normalisation of this feature.

1) Sampling techniques: Additionally, for some models, oversampling and undersampling techniques were applied. The data can be balanced by two simple techniques, undersampling the majority class or oversampling the minority class. In the first case, one option that arises is to cluster the larger class in the same number of clusters as the number of examples of the minority class and take the centroids as training examples of the majority class. In our problem, the number of examples of the positive class is exaggeratedly reduced, therefore, a priori the results will be poorer when undersampling the majority class. On the other hand, a simple oversampling of the examples from the minority class with replacement can lead to overfitting on these examples. One way to avoid this is to create synthetic data of the minority class by combining examples that are close in the space (SMOTE [4]), an approach based on KNN algorithm. There is also a modified technique (MSMOTE [12]) in which the minority examples are labelled according to how they are distributed in the features space, thus we can avoid oversampling noisy examples.

C. Hyperparameter Search

A large number of experiments, using the random search approach to selecting values for hyperparameters were carried out to find good model hyperparameterisations. This involves randomly selecting a value for hyperparameters (for example learning rate) from an appropriate finite set (which may be continuous in the case of learning rate, or discrete in the case of the number of layers in a network). Random search, rather than an exhaustive grid search method was used since the former tends to find reasonable solutions more quickly, especially in the case when some of the hyperparameters have very little influence on performance [2].

D. Scoring

In skew datasets, assessing the performance of a machine can be done using accuracy but it can be misleading: in a set where there are 990 negatives and 10 positives, a machine which always predicts the negative class will be 99% accurate despite misclassifying all the positive samples. To take account of the skewness of the data, and thereby balance classifying both majority and minority classes equally, we used the F_β score, which is a weighted harmonic mean of precision and recall, in other words, a score that weights the importance of false positives or false negatives in the prediction. The formula is

$$F_\beta = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + \beta^2 \cdot FN + FP}. \tag{1}$$

For $\beta > 1$, more consideration is given to the recall component of the score, in other words, correctly identifying positives becomes more important than misclassifying negatives. For $0 \leq \beta < 1$ the inverse is the case. Using F_β score, with $\beta > 0$, as a metric for model performance is especially useful when correctly identifying one class (typically the minority class) is more important than misclassifying the other class. For example, in the case of fraudulent transactions, the cost of investigating a false positive result (a valid transaction incorrectly classified as a fraud) is likely to be less than the costs associated with missing a true fraudulent transaction.

A general measure of recall and precision performance of a classifier can be determined by estimating the area under what is known as the precision recall curve as β increases from 0 to infinity. However, because of the small number of positive samples in each validation set (25 points), the variance, over all validation sets, of the area under the curve (AUC) tends to be quite high, making performance comparisons using this method difficult. For this reason, instead of comparing the average AUC for each machine, the F_β score was calculated for discreet values, specifically $\beta \in \{1, 10, 50\}$. These values were selected to give a realistic range of relative importances of true positives and false negatives in a fraud identification use case. Averages for each of these three scores were calculated using the results from the cross-validation process.

IV. Problem framing

The specific problem addressed in this report was to identify which observations correspond to the fraudulent class and which to the authentic transaction class. On the surface, this may seem like a basic binary classification task, however, there are different ways of conceiving the data distributions, which lead to different approaches. In this section, we describe three perspectives from which the problem can be considered.

A. Anomaly Detection

One can consider the problem as an anomaly detection task. Methods used for anomaly detection are trained only on the majority class (observations considered *normal*) and learn its distribution. Having trained a model, observations from either class can be shown to them and they will reject as anomalous data points which do not meet the definition of normal.

In a fraud detection setting, if one assumes that there is a dense collection of valid normal transactions, with sets of fraudulent transactions on the periphery. Given an appropriate treatment, it may be possible to construct a manifold which neatly bounds the valid transactions.

B. Binary Classification

In a binary classification problem, one assumes there are two different classes of observations which come from two different distributions[2]. Both classes are distributed in a high-dimensional space, and points are withdrawn for them. The goal then is to find and define the borders which bound the regions in which one class is more likely to be found than the other. This can be achieved by typical discriminative algorithms, such as Bayes classification, Multi-Layer Perceptrons, Support Vector Machines.

C. Multi-class Problem

How one selects the class labels for the known set of observations is an important step in the creation of a classification model. In the most basic case, one may choose to use two labels, positive and negative, without regard to distributions of these classes in the feature space. Perhaps these labels are chosen for convenience, for example, to satisfy some operational business requirement, such as all fraudulent transactions must be recorded and investigated. However, ignoring the feature space means risking the possibility that there are different types of fraud, and consequently defining the problem in such a way that a binary classifier might find it difficult to find separating hyper-surfaces which collect all of the various types of positive on one side, and all the negatives on the other.

Such a situation was found to exist with the dataset used in this paper. Figure 3 shows low dimensional representations of the data, projected using a technique called t-distributed stochastic neighbour embedding (t-SNE). The t-SNE technique works by projecting the data into a low dimensional space, whilst preserving only local distances between neighbouring observationst-SNE [18]. Unsupervised clustering was then performed on this projection using the density-based spatial clustering of applications with noise (DBSCAN) algorithm

[2]We assume that fraudulent transactions are correctly labelled and have inherently different measurable properties.

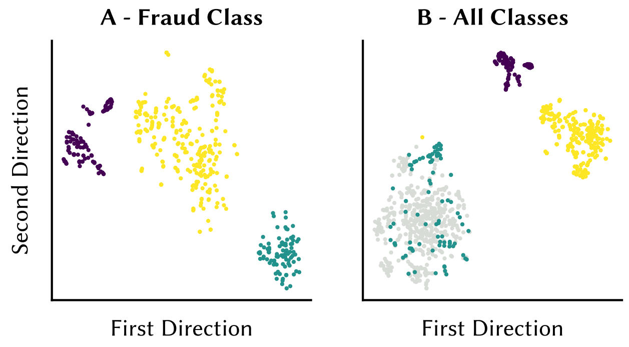

Fig. 3. **A** - A 2D t-SNE projection of fraudulent observations, clustered using the DBSCAN algorithm. **B** - Another 2D t-SNE projection, this time with non-fraudlent data shown in grey (cluster colouring carried over from **A**).

to identify subclasses. The DBSAN algorithm works by first assigning a cluster to each observation, and then repeatedly iterating over the observations, combining clusters which are within some distance ϵ together until no more clusters are within ϵ of one another [9]. Unlike the K-means algorithm, the number of clusters is not defined a priori.

Figure 3 A suggests that the fraudulent class, as defined in the dataset, may in fact, consist of approximately three subclasses, with a majority subclass shown in yellow and two smaller subclasses shown in purple and green. When these subclass labels are then used to colour the fraud observations in figure 3 B, we see that only the green fraud subclass overlaps with the valid transactions coloured in grey, whilst the other two subclasses are well separated. Upon further investigation, it was found that the green subclass tended to have amounts and transaction times typical of the non-fraudulent transactions, whereas in contrast to the other two classes which tended to have unusual small whole number amounts and often occurred at atypical times such as in the early morning. We hypothesise therefore that the green subclass may be duplicate transactions, or opportunistic crime, in contrast to the other classes which may be representative of organised crime from locations to the east of Europe, due to the time differences.

V. Classification Models

In this section, we describe the various models, sampling techniques, and other enhancements which were applied to the problem. A brief description of how each model is given, along with information on important hyperparameters.

A. Autoencoder

Autoencoders (AEs), also known as replicator neural networks (RNNs), are a type of neural network whose aim is to learn an approximation to the identity function: given an input, the target output is a vector as similar as possible to the input $f(x) \approx x$. The architecture consists of an input and output layer of equal dimensions, and a varying number of lower dimension hidden layers. It has two main parts: the encoder and the decoder. The encoder takes the input and compresses it into a lower dimension representation; the decoder tries to reconstruct the input from the compressed data. By reducing

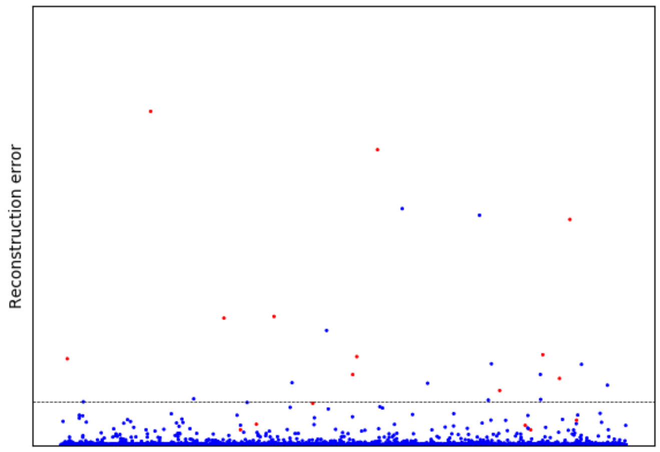

Fig. 4. Reconstruction error. Blue dots represent non- fraudulent transactions and red dots fraudulent transactions. The horizontal doted line is the defined threshold. The data is uniformly distributed over the X axis.

the dimension(s) of the hidden layer(s), the learning of the identity function is non-trivial. The reconstruction error is the difference between the input and the output $||x - x'||^2$, and by minimising it, the AE learns a generalisation function of the data.

AE can then be used for outlier/fraud detection [6], [10]: the network is trained on non-fraud data, minimising the reconstruction error, building a model of the normal data. Then, when presented with new data, the reconstruction error is an indicator of how well the data fits the learnt model, where high values are expected for fraudulent data.

For this exercise, the AE has been trained in two steps. In the first step, the autoencoder is fed with the corresponding training split, where all the data from the positive class are discarded. The reconstruction error is minimised over several epochs using the Adam Optimisation Algorithm. The second step is used to define which value of the reconstruction error is going to be the threshold for classifying new data into the negative or the positive class. The reconstruction error is calculated for all the values of the training set (this time including the positives) and then the threshold is chosen to maximise the F-score. Figure 4 shows the reconstruction error for a single validation set, the horizontal line represents the threshold.

The hyperparameters that have been tuned using a random search are the number of hidden layers, the number of units per layer, and the activation functions used in the hidden layers.

B. Support Vector Machines

Support vector machines (SVMs) are a collection of methods, so called since they define classification boundaries in terms of a non-negatively weighted combination of the observations' feature vectors when evaluated in the dual form. SVMs are useful when dealing with relatively small datasets ($\sim$10000), but become impractical for large datasets, such as the one in this study, since their time complexity is $O(P^3)$. For this reason, the sampling techniques described in section III-B1 were used before training the models. In all cases, values for important hyperparameters, correlated with high performance, were identified using the random hyperparameter search mentioned in section III-C.

1) Binary Support Vector Classifiers: Binary support vector classifiers (B-SVCs) are popular models for binary classification problems [1]. This method maximises the margin between positive and negative classes. When the classes are not linearly separable in feature space, two main strategies can be applied: adding some slack to the margin thereby allowing for any point k to violate the margin by a distance (ϵ_k), and projecting the data to a higher dimension feature space, using a non-linear function (ϕ), in an effort to make it linearly separable. Finding the maximum margin is then equivalent to solving the quadratic optimisation problem

$$\max \frac{1}{2}\|\bar{w}\|^2 + C\sum_{k=1}^{P} \epsilon_k \text{ s.t. } y_k(\bar{w}^T\phi(x_k) + \bar{b}) \geq 1 + \epsilon_k \quad (2)$$

where C is a parameter chosen beforehand, which weights the importance of the margin violations. Various non-linear transformations were evaluated, specifically those corresponding to polynomial, radial basis function (RBF), and sigmoid kernels. In these cases, the dual form of the problem is solved.

Furthermore, because the training sets are quite large for SVM methods, different approaches were used for selecting smaller subsets of data, as described in section III-B1.

2) Multi-Class SVCs: Multi-class SVCs (MCSVCs) are an extension to SVCs, which use a one-versus-one voting system, first described by Knerr et Al. in the context of training neural networks [14]. For a problem with K classes, $K(K1)/2$ SVC machines are trained, corresponding to the total number of pairwise combinations of classes. Each machine casts a vote, and the class with the greatest number of votes is given as the prediction.

This method was used in combination with the subclasses generated by the clustering procedure described in section IV-C.

3) Support Vector Data Descriptors: Support vector data descriptors (SVDDs) are designed with anomaly detection in mind, as such, they are trained using only a single class, typically the majority class [17]. Such methods are useful when no examples of anomalous data have been identified, but a need for detecting potential outliers still exists. To train the model, after normalising the data, an n-sphere of radius R and dimension equal to that of the feature space, which minimises

$$F(R) = R^2 + C\sum_{k=1}^{N} \epsilon_k, \quad (3)$$

is fit to the data, where N is the number of observations and C and ϵ are defined as before. This is done subject to the constraint that

$$||\mathbf{x}_k|| \leq R^2 + \epsilon_k,\ \epsilon_k \geq 0\ \forall i \quad (4)$$

where $\mathbf{x}_k$ is the k^{th} observation. After training the model, new observations are classified as outliers if they lie outside of bounding n-sphere. As with the other SVMs methods, undersampling was used to make the problem tractable.

C. XGBoost

XGBoost, short for *Extreme Gradient Boosting*, is an efficient gradient boosting library used for machine learning supervised problems, both in classification and regression [5]. It is framed in the family of machine learning tools known as ensemble methods, which attempt to reduce the variance and improve generalisation by averaging over different learning machines. This trick is particularly useful for machines that can be trained quickly, and hence decision trees or linear models are preferred options for ensemble learning and, particularly, for XGBoost.

1) CARTs: Decision trees are binary trees, with each binary division corresponding to a rule based on a single feature. The choice of this rule is subject to entropy maximisation of the leaf nodes. XGBoost uses a kind of decision tree known as CART (Classification And Regression Trees). CARTs not only include the decision rules but also assign a score to every leaf node. Such scoring, along with the fact that they depend strongly on early decisions (i.e. they have high variance), also reinforce their suitability for ensemble methods.

The process of choosing both feature and splitting point for each decision at every node has three steps: 1) for each feature, the n data-points are sorted by the feature value, 2) a linear scan is utilised for finding the best split along the feature, and 3) the best splits along the d features are compared and the best feature is selected. The time complexity for building a tree of depth K is $O(d\,K\,n\,log\,n)$, although it can be improved by approximation or caching [5].

2) Boosting: The expected variance of n random independent variables is proportional to 1/n, a theoretical idea that supports ensemble methods power for improving generalisation. However, this requires the aggregated machines to be independent, to learn different things. When some correlation ρ exists between random variables, we have instead

$$\mathbb{E}\left[\frac{1}{n^2}\left(\sum_{i=1}^{n} X_i - \mu\right)^2\right] = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

, with the left term remaining irreducible and proportional to ρ. Boosting techniques aim at reducing the correlation factor by building new trees one at a time (additive training), and training them on the areas where other trees did not learn well, i.e. on the residual errors. In order to ensure de-correlation, individual trees are very shallow (less than 10 levels deep), and perform slightly better than chance (weak learners).

As it is the case in most machine learning techniques, boosting models could arbitrarily increase their complexity and over-fit the data, so regularisation techniques must be applied, such as:

- Early stopping when performance against a validation set starts deteriorating.
- Dropout on trees [19][3]
- A regularisation term, γ, that imposes penalties on the amount of leaves a tree has; and a L2 regularisation term, λ, that limits the weights on the leaf nodes of the CARTs.

[3]Note that this method cannot be used at the same time with early stopping.

The combined regularisation term has the form

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

In order to find (tune) the hyper-parameters that produce the best machines for our problem, random search over the parameter space is carried out; with the decision being made over the results of the trained machines on a validation set extracted from the training set. Different sets of hyper-parameters are tuned for every one of the three chosen beta scores.

D. Multilayer Perceptron

We follow another approach based on a multilayer perceptron (MLP) as a classifier, that is to say, a fully connected feed-forward neural network with one or more hidden layers between input and output layer. Each unit acts as an activation function that depends on the inner connections. In this way, we can recognise non-linear patterns between the inputs and the outputs.

The main problem with this model lies in the optimisation of the connection weights trying to not overfit to the training set and thus achieve good generalisation results. The error is measured by the entropy between the positive and negative class. In addition, Adam or adaptive moment estimation [13] is used as training algorithm, this method works very efficiently for different architectures of deep neural networks. It is a stochastic optimisation method that only needs to compute first-order gradients. We start from the gradient of the output error:

$$g_t = \nabla_w J_t \tag{5}$$

Then we compute adaptive estimates of the first (mean) and second moment (un-centred variance) of the gradient for each iteration t. The parameters β_1 and β_2 indicate the exponential rate of decay of the moments estimates. Moreover, the initialisation bias of these estimates is corrected.

$$m_t = [\beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t] \cdot \frac{1}{1-\beta_1^t} \tag{6}$$

$$v_t = [\beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2] \cdot \frac{1}{1-\beta_2^t} \tag{7}$$

The moments allows us to set a different adaptive update for each of the weights analogous to simulated annealing, hence, Adam works well with sparse gradients.

$$w_t = w_{t-1} - \alpha \cdot \frac{m_t}{\sqrt{v_t}+\epsilon} \tag{8}$$

The units used are ReLU, which are computationally faster than the traditional activation functions, they could also save part of the vanishing gradients in deeper architectures [15]. In addition, we impose a penalty on the model complexity with L_2 regularisation, it means a weight decay before the updates. The training is also accelerated by iterating through mini batches of the dataset.

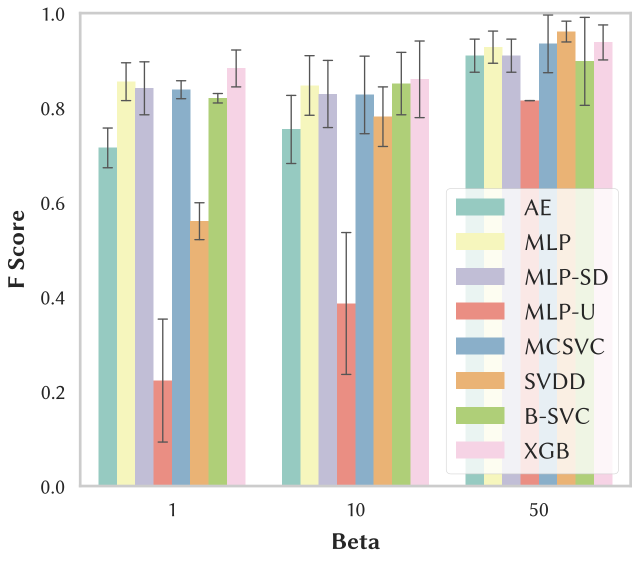

Fig. 5. Mean F_β scores shown with standard deviation over 10 splits for $\beta \in \{1, 10, 50\}$.

The MLP classifier is contrasted in different situations to see if we improve the results derived from the imbalanced classes problem. First, the base training examples are used. On the other hand, in order to balance the classes, we generate new examples of the positive minority class. On the contrary, a K-means clustering of the majority class is done, to reduce the number of examples of the positive class to the centroids of the cluster. In addition, the clusters from the t-SNE projections of the minority class are used to label the training examples and treat the fraud detection as a multi-class problem.

VI. Results and Analysis

Before comparing the performance scores of the different models, it is important to examine the spread of the F_β score results for each model across the validation splits (see Table I for standard deviations). From Figure 5, we see that there is a considerable overlap of plus/minus one standard deviation from the mean for the top performing models for each value of $beta$ making it impossible to draw very reliable conclusions on which model performs best. Should the spreads have been smaller, then for any pair of models, a test of significance suitable for comparing cross-validated results, such as Wilcoxon signed-rank test, could be applied to estimate the probability of the observed difference between the pair, under the null hypothesis $\mathcal{H}_0$ that both models have the same mean performance [7]. However, in this case, no such test is necessary to reach the aforementioned conclusion given the significant overlap of results.

The high variance is likely a consequence of the low number of positive samples included in the test set (as described previously, the validation set in each split contains only 25 positives samples). Misclassifying a single sample of the 25 accounts for a 4% error in the prediction.

Having said this, it can be seen from Figure 5 that the majority of the models perform similarly, with the exception of the MLP with undersampling (MLP-U) which had a fairly poor performance. The poor MLP-U for $\beta = 1$ performance may have been a result of reducing the number of training examples too far, meaning only very small networks, with high bias, can be trained to generalise properly. This effect was observed to be quite extreme, and the best MLP-U machine contained only a single hidden neuron, equivalent to logistic regression. Of note is the highest scores were achieved for $\beta = 50$, where the score is bounded since we can achieve apparent good results by classifying all observations as positives ($F50 = 0.815$).

	Model	TP	FP	Prec.	Rec.	F avg	F std
F1	AE	17.8	6.6	0.734	0.712	0.715	0.042
	MLP	20.0	1.7	0.925	0.800	0.855	0.040
	MLP-SD	18.7	0.7	0.965	0.748	0.841	0.056
	MLP-U	5.4	1254.9	0.370	0.216	0.223	0.130
	MCSVC	20.7	3.7	0.858	0.826	0.838	0.019
	SVDD	11.3	4.0	0.742	0.453	0.560	0.039
	B-SVC	19.7	3.3	0.862	0.787	0.820	0.010
	XGB	20.2	0.5	0.977	0.808	**0.883**	0.039
F10	AE	19.4	77.9	0.237	0.776	0.754	0.072
	MLP	21.5	41.8	0.546	0.860	0.847	0.063
	MLP-SD	21.1	51.2	0.518	0.844	0.829	0.071
	MLP-U	18.6	4994.5	0.022	0.744	0.386	0.150
	MCSVC	20.7	3.7	0.859	0.827	0.827	0.082
	SVDD	22.0	320.3	0.064	0.880	0.781	0.063
	B-SVC	21.3	10.3	0.686	0.853	0.851	0.066
	XGB	21.8	37.0	0.528	0.872	**0.860**	0.081
F50	AE	24.1	3708.4	0.006	0.964	0.910	0.035
	MLP	23.9	2250.7	0.010	0.964	0.928	0.034
	MLP-SD	23.8	2991.9	0.059	0.952	0.910	0.035
	MLP-U	25.0	14215.8	0.002	1.000	0.815	0.000
	MCSVC	23.7	782.3	0.029	0.947	0.935	0.061
	SVDD	24.7	1685.0	0.014	0.987	**0.961**	0.022
	B-SVC	23.0	1496.3	0.019	0.920	0.898	0.093
	XGB	23.9	1231.4	0.044	0.956	0.938	0.037

TABLE I
Average precision, recall and F_β scores of the models over the ten training/validation sets. Top performing model shown in bold.

The support vector data descriptors (SVDD) performed poorly for $\beta = 1$, typically when $\beta = 10$, and best in class when $\beta = 50$. This can be explained by the influence of C on the behaviour of $F(R)$ from 3. When $\beta = 1$, violations of the n-sphere boundary (false positives) cause a significant effect of the F_1 metric, and so C must be set quite high to balance false positives and false negatives. However, when $\beta = 50$, false positives are far less important, and so C can be chosen to be smaller, resulting in a smaller radius R of the bounding sphere. As a consequence, almost all of the anomaly (positive) class will then lie outside of the sphere, resulting in a very high recall, and a very high F_{50} score.

XGBoost performs slightly better than the rest on F_1, and among the top models for F_{10}. One could argue that the reason for this is its high generalisation ability, a feature that is critical for a case with so few data-points from one of the classes. For F_1, it can be seen that the precision is indeed close to 100%, just leaving aside a few of the points (presumably the ones that belong to the 'mingled' class).

AE's best performance on F_1 and F_{10} scores was achieved with a relatively low compression of the data (reducing the feature vector from 31 inputs to 26-28 in the compressed layer). This is in line with the fact that the original data had been pre-processed, extracting the principal directions. However, the average performance was low compared to other methods, probably due to the difficulty of finding an identity function that describes such a large data-set. For the F50 score, a higher compression was possible (down to 16 hidden units), since a more abstract representation is possible: finding a high number of true positives outweighs the fact of miss-classifying negative samples.

To deal with the small quantity of positives, synthesising new data from the positive class was also applied in conjunction with the different models. However, this technique didn't seem to add any relevant improvement in the performance on the test set. As shown in the t-SNE analysis, the fraudulent transactions are clustered in different areas of the feature space. One of these classes tends to overlap with the negative class, thus we believe that SMOTE could be adding noise examples in this direction. The modified version of SMOTE that takes into account frontier and noisy points should have been explored.

VII. Conclusion

In this paper, we explored different algorithms for detecting fraud in the Kaggle Credit Card dataset. We used the F_β score to compare their performance. This allowed us to play with the sensitivity to false positives on the prediction. From the results described in the previous section, we can conclude that XGBoost will have the best-expected performance when presented with new data drawn from the same distribution as the dataset under study. However, as it was mentioned earlier, this must be caveated due to the relatively high variance in the score results.

Given the difficulty discerning which model had the highest performance, as a result of the broad confidence intervals, in future work, it would be useful to evaluate the models using splits with more balanced sizes of training and validation sets, in an effort to reduce the variance observed over the validation splits. It is likely that several ratios of train versus validation size would need to be evaluated, to find a good balance between worsened mean F_β scores (a result of less training data) and reduced variances.

It is worth highlighting that implementing an algorithm that could consistently detect all fraudulent transactions is something that cannot be achieved with the given dataset since sometimes their features are indistinguishable from normal transactions (green group in Figure 3). As theorised earlier, if these transactions are simply opportunistic theft some additional discriminative features may be required, for example, the cardholders proximity to the transaction if it was made in the real world, using mobile phone position for example.

The results also show that it may be possible to improve performance by first identifying if there exist sub-classes of fraudulent transactions, which correspond to different kinds of criminal activities and training a multi-class classifier as appropriate. However, this approach alone would not be sufficient in our opinion to protect against fraudulent activity, as it is conceivable that in the future, new fraudulent techniques could be developed which may be undetected by previously trained multi-class methods. We, therefore, recommend that any binary or multi-class approach be supplemented with an anomaly detection model, with the goal of identifying novel fraudulent behaviour. By implementing this combination of models, financial institutions would be able to minimise both known and unknown risks.

References

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying Support Vector Machines to Imbalanced Datasets. *LNAI*, 3201:39–50, 2004.

[2] J. Bergstra JAMESBERGSTRA and U. Yoshua Bengio YOSHUABENGIO. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[5] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[6] H. A. Dau, V. Ciesielski, and A. Song. Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class. In *Simulated Evolution and Learning*, volume 8886, pages 311–322. Springer, Cham, 2014.

[7] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[8] D. L. Donoho. Aide-Memoire. High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality. *American Math. Society Lecture-Math Challenges of the 21st Century*, pages 1–33, 2000.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[10] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180, 2002.

[11] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[12] S. Hu, Y. Liang, L. Ma, and Y. He. Msmote: improving classification performance when training data is imbalanced. In *Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on*, volume 2, pages 13–17. IEEE, 2009.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

[15] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[16] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166, Dec 2015.

[17] D. M. Tax and R. P. Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[18] L. Van Der Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[19] R. K. Vinayak and R. Gilad-Bachrach. DART: Dropouts meet Multiple Additive Regression Trees. In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 489–497, San Diego, California, USA, 09–12 May 2015. PMLR.

[20] K. Worobec. Fraud - The Facts 2017. Technical report, Financial Fraud Action UK Ltd., London, 2017.