Anomaly Detection: Supervised vs Unsupervised Techniques

DAT550 Group Project

Alireza Hossein Zadeh Nik
University of Stavanger, Norway
a.hosseinzadehnik@stud.uis.no

Prava Thapa
University of Stavanger, Norway
prav.thapa@stud.uis.no

Sondre Tennø
University of Stavanger, Norway
s.tenno@stud.uis.no

Håkon Teppan
University of Stavanger, Norway
h.teppan@stud.uis.no

ABSTRACT

Nowadays, businesses or even consumers prefer to invest a great deal of time and effort in finding patterns that do not accord with what is considered normal behaviour instead of losing their money and assets due to abnormal events. This is referred to as anomaly detection, the process of detecting data instances that significantly deviate from the majority of data instances. Anomaly detection has been an active research area within diverse application domain over the past few years. One of the biggest challenges in many anomaly detection applications is dealing with highly imbalanced data instances, causing the learning algorithms to experience different complexities.

This project aims to study different ML algorithms to tackle anomaly detection problem on a real-life dataset of credit card transactions to detect fraudulent ones. We have grouped different anomaly detection techniques into two major categories: Supervised and unsupervised anomaly detection techniques. Due to the availability of labels, different supervised algorithms such as logistic regression, Random forest, NB classifier, and K-nearest neighbour was tested on a balanced and imbalanced dataset. We also experimented with multiple unsupervised techniques including isolation forest, one-class SVM, elliptic envelope, k-means clustering, Auto-Encoder, local outlier factor and Markov chain on the dataset since obtaining the labelled dataset for the majority of real-life applications is challenging and impractical. The performance measure for each model was evaluated and discussed thoroughly.

KEYWORDS

Normal, Anomaly, Supervised and Unsupervised, Auto-Encoder, Isolation Forest, One-Class SVM, Local Outlier Factor, Elliptic Envelope

1 INTRODUCTION

A common need when analysing real-world datasets is determining which data points stand out as being dissimilar to all others [6]. Such instances are known as anomalies and the goal of anomaly detection is finding patterns in data that do not conform to expected behaviour [7]. These patterns are often referred to as anomalies, outliers, abnormalities, rare events, deviants, peculiarities or contaminants in different application domains. The capability to detect anomalous behaviour can provide useful insights across different industries and applications such as fraud detection for credit cards, IT analytics, network intrusion systems, medical diagnostics and military surveillance for enemy activities. As an example, which our study case is based upon, it is critical for companies working within the financial industry to quickly and correctly identify and react to fraudulent transactions. Fraud could be cast as a deviation from normal transaction data and defined as any criminal activity that corresponds to high payments, purchase of items never purchased by the user before, high rate of purchase, etc, which could result in financial loss to a cardholder and personal gain of the fraudsters [20]. Thus, it is the task of anomaly detection methods for the recognition of these fraudulent activities and thus preventing them. Anomaly detection is affected by unwanted noise in the data. We need to remove the noise or any unwanted objects before any analysis is performed on the data because the noise acts as a hindrance to data analysis.

Machine learning techniques are primary methods in the detection of anomalies and outliers. These techniques, in terms of the availability of labelled data, could be divided into two groups: supervised and unsupervised techniques.

In supervised techniques, a model is trained on a data set that has labelled instances for normal as well as anomaly class to classify new instances. There are two major issues in supervised anomaly detection. First, obtaining accurate and representative labelled data is expensive and challenging as the training data set is often labelled manually by a human expert and it requires plenty of effort [7]. Secondly, the training dataset is highly imbalanced in most cases, meaning that the anomalous instances are far fewer compared to the normal instances. With sufficient normal and anomalous instances, the supervised anomaly detection problem is like a classification task where the machine can learn to accurately predict whether a data point is an anomaly or not.

On the other hand, unsupervised techniques do not require labelled training data, and thus are most widely applicable. The techniques in this category detect outliers solely based on intrinsic properties of the data instances and make the implicit assumption that the normal instances are far more frequent than anomalies in the test data. However, the nature of the anomaly is often highly specific and the anomalous behaviour may change over time [17].

Supervised by Vinay Setty.

Thus, fully unsupervised techniques could detect anomalies that correspond to noise and may not be of interest. In many practical cases, the labels of normal instances are far easier to obtain than anomalies. Therefore machine learning methods would be trained on normal instances so that they can differentiate between upcoming normal and anomalous points.

In this group project, we studied different supervised and unsupervised techniques on a real-life transactions dataset from an international credit card corporation. Due to the imbalanced nature of the dataset, different kinds of under-sampling and over-sampling methods were conducted to make the training set balanced and their effect was compared for different supervised learning techniques such as random forest, logistic regression, K-nearest neighbour and NB classifier. For unsupervised learning, we experimented with different algorithms like Isolation Forrest, One-class SVM, deep Auto-Encoders, local-outlier factor, etc, and model performances were compared based on different metrics such as precision, sensitivity, and area under the ROC curve. The rest of this paper is organized as follows:

- Different challenges of anomaly detection problem
- Data description and Visualization
- Theory and Method
- Performance measure
- Experimental results

2 DIFFERENT CHALLENGES OF ANOMALY DETECTION

Anomaly detection brings distinct challenges and complexities compared to the majority of analytical and learning methods. This section summarizes such challenges as follows.

2.1 Class imbalance

Anomalies are typically rare data instances. Therefore, it is difficult to collect a large amount of labelled anomalous instances. This results in a highly imbalanced (skewed) data set. As a result, machine learning techniques tend to produce unsatisfactory predictions for the minority class. For imbalanced data set even the best learning algorithms are incapable of detecting anomalies from normal instances and they would identify too many false positive i.e., normal points as anomalous ones. There exists a handful of techniques for converting an imbalanced dataset to the balanced one.

2.1.1 ***Random Under Sampling.*** Over-sampling and under-sampling techniques involve introducing a bias to select more samples from one class than from another, to compensate for the imbalanced nature of the dataset. The simplest approach is to choose examples from the transformed dataset randomly, called random resampling. We define Random Sampling as a naive technique because it makes no assumptions of the data beforehand. Random under-sampling involves randomly choosing examples from the majority class and deleting them from the training dataset. This process is simple to implement with fast execution time and it is desirable for large datasets. The biggest limitation of the random under-sampling method is that the deleted examples from the majority class may have a critical impact on what the outcome of the algorithm would be [26][15][5].

2.1.2 ***SMOTE.*** Synthetic Minority Oversampling Technique is a type of data augmentation technique to oversample the dataset with new synthesized data points from the existing dataset. To oversample a subset of the dataset, we take a sample from the dataset, and consider its k-nearest neighbours. To create a synthetic data point, we take the vector between one of those k-neighbours, and the current data point. Then we multiply this vector by a random number x which lies between 0 and 1. The result obtained is added to the current data point to create a new synthetic data point. This procedure will then be used to create as many synthetic data points for the minority class as we want until we get the desired distribution. A limitation to this technique is that synthetic examples are created without considering the majority class. This could result in ambiguous examples if there is a strong overlap of the classes [5][16][8].

2.1.3 ***ADASYN.*** The adaptive Synthetic Sampling Approach uses a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those minority class examples that are easier to learn. ADASYN improves learning for the data distributions in two ways. At first, it reduces the bias we get from the imbalance in our dataset and secondly it adaptively shifts the classification decision boundary towards the difficult examples. A limitation for ADASYN is that the data is generated in neighborhoods with high amounts of the majority class can potentially generate a lot of false positives [14][25][12].

2.2 Output of Anomaly Detection

An important aspect for any anomaly detection algorithm is the way in which the anomalies are detected. Generally, the outputs are reported in two ways:

2.2.1 ***Anomaly Score.*** This technique assigns a score to each instance based on the degree to which that instance is considered anomaly [7]. In other words, data points are ranked based on an anomaly score and a specific threshold. In general, anomaly scores reveal more information than binary labels.

2.2.2 ***Labels.*** Some techniques assign a label as normal or anomalous to each data points. These techniques do not allow us to choose directly by defining anomalies as compared to anomaly scores. However, we can control them indirectly through parameter choices within each technique [7].

2.3 Low recall rate and high False positive

It is difficult to detect all the anomalies since they are highly rare in the dataset. Many sophisticated algorithms suffer from high false positives and low recall, meaning numerous normal instances are wrongly reported as anomalies when true anomalies are lost. One of the most difficult challenges for anomaly detection algorithms is how to reduce false positives and improve recall rates [6].

2.4 Detection of complex anomalies

Most of the existing anomaly detection algorithms are for point anomalies, which cannot be used for the other two kinds of anomalies, namely, contextual anomaly and collective anomaly as they have completely different behaviours from point anomalies. Current techniques mostly focus on anomalies from single data sources, while many applications require the detection of anomalies from multiple data sources, e.g., multidimensional data, graph, text, etc [6].

3 DATA DESCRIPTION AND VISUALIZATION

The dataset contains information of transactions made by European cardholders within two days in September 2013. Out of 284,807 transactions, only 492 (i.e., 0.172%) are fraud cases. There are 31 features in this dataset, where V1, V2, …, V28 are transformed using PCA to preserve confidentially. Other 3 feature that do not bind with PCA are "Time", "Amount", and "Class". The "Class" is binary feature with 1 indicating of fraud and 0 indicating of normal transactions.

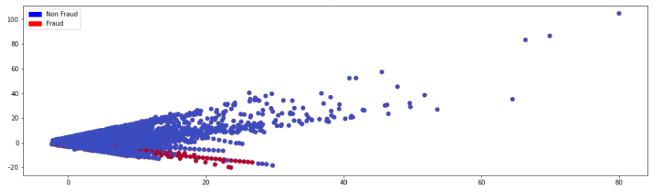


Figure 1: PCA on the original imbalanced dataset

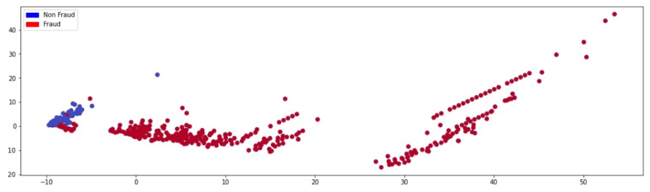


Figure 2: PCA on the balanced dataset

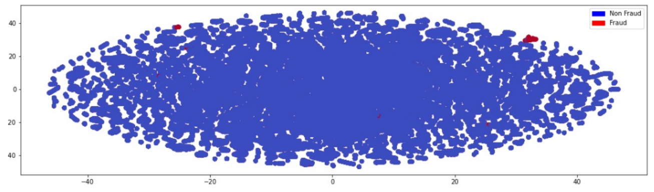


Figure 3: t-SNE on the original imbalanced dataset

After the pre-processing stage, we try to visualize the dataset to get some intuition about the nature of anomalies and data instances. We implemented *PCA* (principal component analysis) and *t-SNE* to visualize both imbalanced dataset and balanced subset of it. Through t-SNE visualization of the imbalanced dataset, we can see that the fraudulent cases (the anomalies) mostly gather at two places as seen in figure 3. Given that most of the anomalies are clustered together, we can assume they are very similar to each other which means they have some distinct value in the features which makes them cluster together.

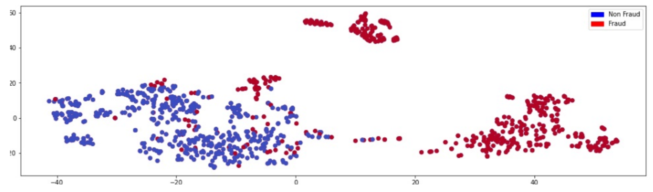


Figure 4: t-SNE on the balanced dataset

We then apply t-SNE on balanced dataset. This dataset contains equal number of normal data points and anomalies i.e., 473 normal data points and 473 anomaly points. The normal data points are chosen randomly from the dataset while we use all of the anomalies. After t-SNE application on the balanced dataset, we can see a big difference between the anomalies and the normal data points. The normal data points are gathered by themselves with a minority of anomalies clustered with them and majority of them are far away in the feature space which can be seen in figure 4. Although the subsample is small, given the fact that original dataset is imbalanced, both PCA and t-SNE perform relatively accurate. Both of them cluster the anomalies and the normal data points in separate regions. This gives us an indication that separating the anomalies from the normal data points has a high chance of performing well when we balance the dataset.

4 THEORY AND METHOD

This section describes different unsupervised methods applied in this project to identify anomalies.

4.1 Auto-Encoders

Auto-encoders are a special kind of neural networks that can discover low-dimensional representations of high-dimensional input data and then reconstruct that input data from the output [2]. They consist of a pair of two connected sub-networks: an *encoder* that learns to map high-dimensional input data to a dense, low-dimensional representation (termed the bottleneck- latent dimension), and a *decoder* that learns to map this low-dimensional representation back to the original input data (figure 5) The goal of such a process is learning an efficient *compression* function that maps input data to a salient lower-dimensional representation [17] using an encoder and successful reconstruction of the original input data using the decoder. If the number of neurons in the hidden layers is less than that of the input layers, the hidden layers will extract the most relevant information and learn to preserve the essential pattern of the input values and ignore the irrelevant parts like *noises*. Consequently, in an auto-encoder architecture, the hidden layers must have fewer dimensions than those of the input or output layers. If the number of neurons in the hidden layers is more than those of the input layers, the neural network will be given too much capacity to learn the data [3]. In an extreme case, it could

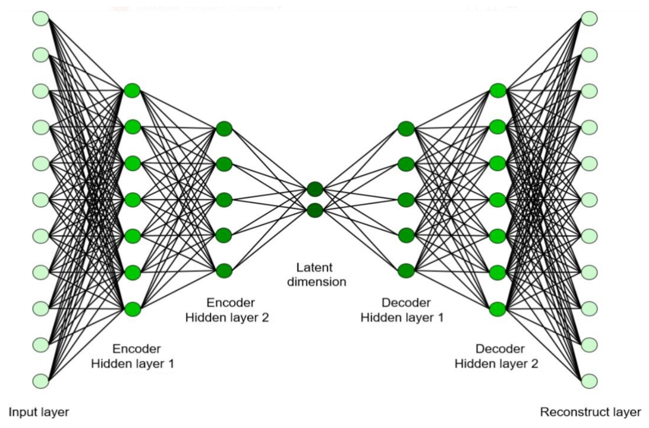


Figure 5: Illustration of Auto-Encoder structure

just simply copy the input to the output values, including noises, without extracting any essential information. Figure 5 depicts the encoding and decoding process in a deep auto-encoder architecture. The decoding process mirrors the encoding process in the number of hidden layers and neurons. The encoding process compresses the input data to get to the core latent dimension (bottleneck) and the decoding process properly reconstructs that dense information back to the input. Auto-encoder is trained by minimizing the reconstruction error (loss function), which is the difference (mean squared error) between the input and the reconstructed output. Since this model does not require the target variable like the conventional ANN, it is categorized as an unsupervised learning method.

4.1.1 ***Auto-Encoder in anomaly-detection.*** When the neural network goes through the input training data to generate a compressed representation via the hidden layers and fine tunes the weights of all the hidden layer nodes, the weights will represent the kind of input they have observed in the training process. As a result, if we try to input some other type of data significantly different from the data it has seen during training, auto-encoder will typically not succeed at reconstructing data. This property is particularly useful for the task of anomaly detection. For this purpose, we train the auto-encoder on normal data samples in a semi-supervised manner in the training process. So, when the model is fed new samples, it yields small reconstruction error small for normal data samples and large for abnormal ones(figure 6). By using the reconstruction error as an anomaly score, we can flag anomalies with reconstruction errors above a given threshold.

4.1.2 ***Sparse Auto-Encoder.*** A reliable auto-encoder must make a trade-off between two essential facts [27]:

- Sensitive enough to inputs so that it can accurately reconstruct input data
- Able to generalize well even when evaluated on unseen data

In the auto-encoder model described earlier, representations were only constrained by the size of the hidden layer. In such a situation, the model can be thought of as a more powerful (nonlinear) generalization of PCA. On the other hand, a sparse auto-encoder is

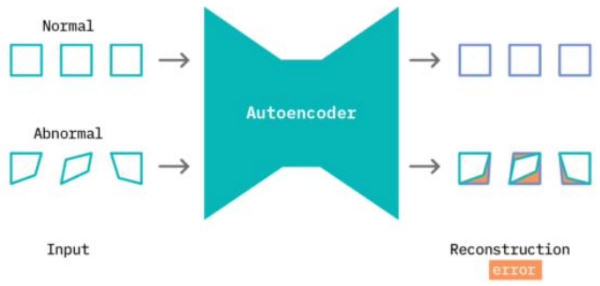


Figure 6: Anomaly Detection in Auto-Encoder [17]

an auto-encoder whose loss function is composed of two different parts. The first part is the mean squared error while the second term would act as a sparsity constraint on the activity of the hidden representations. So that fewer nodes are encouraged to activate when a single sample is fed into the network. This would guarantee that the model is learning useful insights instead of redundant information in our input data. This can be achieved by adding an *activity_regularizer* to the hidden layer. Generally, two different approaches to construct sparsity penalty are L1 regularization and KL-divergence.

4.2 Isolation Forest

Isolation Forest is an unsupervised machine learning algorithm for detecting anomalies. The main principle of Isolation Forest is to isolate the anomalies instead of profiling normal points, which is the most common technique for anomaly detection. Isolating here refers to the way Isolation Forest separates an instance from the rest of the instances because the anomalies are few and differ from the rest and therefore more susceptible to isolation. So the point of the Isolation Forest is to isolate out these anomalies. Isolation Forest is a tree ensemble method and is built based on decision trees. In these decision trees, partitions are created by randomly selecting a feature, and then selecting a split value at random between the min and max value of the feature. Outliers are less frequent than a regular observation and have different values, which means they are further away from the normal observations in the feature space. That is why by using the random partitioning they should be identified closer to the root of the tree(shorter average path), with fewer splits required (figure 7). A normal point would require more partitions to be identified than what an anomaly would. An anomaly score will be calculated, scores close to 1 indicates that the data-point is an anomaly, scores smaller than 0.5 indicate that it is a normal observation [19][23][18].

4.3 One-Class SVM

One-class SVM is an unsupervised learning algorithm that is trained only with good observations from the dataset. The key idea of One-Class SVM is that [13]:

- a kernel is used to map training data from input space to the feature space

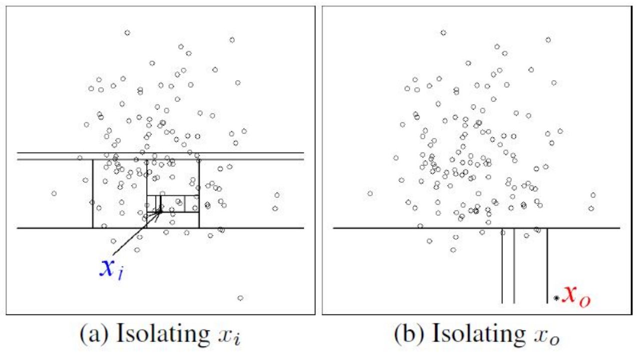


Figure 7: Anomalies are more susceptible to isolation and hence have short path lengths [19]

- a hyper-plane is found with maximum margin in feature space that separates the mapped data from the original data.

All the data points are separated from the origin in feature space F, and the distance to the origin is maximized from this hyper-plane. As a result, a binary function is obtained that captures regions in the input space where the probability density lives. This binary function returns two values: +1 for a small region that captures the training data points and -1 elsewhere [21]. The following quadratic minimization program is used to separate the data points from the origin. When this quadratic minimization programming is derived

$$\min_{w\in F, \boldsymbol{\xi}\in\mathbb{R}^\ell, \rho\in\mathbb{R}} \quad \frac{1}{2}||w||^2 + \frac{1}{\nu\ell}\sum_i \xi_i - \rho$$
$$\text{subject to} \quad (w \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0$$

using a simple kernel and Lagrange technique for the dot product calculation, the decision function is obtained as:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho\right) \quad = \operatorname{sgn}\left(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho\right)$$

This algorithm, therefore, finds a hyper-plane defined by ω and ρ which will separate all the data points from the origin and it will have maximum distance from the origin among all such hyper-planes.

4.4 Elliptic Envelope

Elliptic envelope works by creating an imaginary elliptical area around a given dataset by assuming that all the data are expressions of high dimensional Gaussian distribution [1]. The values that fall under the boundary of the elliptical area are the normal data, i.e., non-fraudulent data with values 0 in our case and the values that fall outside this area are outliers, i.e., fraudulent data with values 1. This algorithm uses a covariance determinant to estimate the size of the ellipse [24]. This covariance determinant selects non-overlapping data subsamples from the original data, computes mean and covariance matrices across each feature dimension in the subsample. It also computes the Mahalanobis distance to measure the distance of each data point from the mean distribution for each feature dimension. This method takes a small portion from the original data, calculates mean, covariances, recalculates the value of sub-sample and continues to do so until the covariance matrix is converged. The covariance matrix forms an ellipse with the smallest determination of all the subsamples. The data points that lie inside the ellipse are normal data or inliers and those that lie outside are outliers.

4.5 K-means Clustering

K-means clustering is one of the simplest unsupervised machine learning algorithms that puts similar kinds of data into one group and discovers the underlying patterns by using a fixed number of clusters in a dataset [10]. Clusters are a collection of data points that have some similarities. This algorithm selects a group of centroids at random as a starting point for every cluster and then optimizes the position of centroids by performing calculations in iteration. This algorithm stops working when the centroids are stabilized or when we achieve a defined number of iterations.

4.6 Local Outlier Factor (LOF)

LOF is also an unsupervised algorithm. It measures the density between the k-nearest neighbours by using Local Reach-ability Density (LRD). Briefly, LRD is a factor that measures the density of each k-nearest neighbour group. This is done by taking the inverse of the sum of the distances between the point in question and its k-nearest neighbours. The value of LOF tells us if it is an outlier or not [4]. If the LOF of a data point is greater than 1, it is an outlier and if it is lesser than 1 then it is normal data. With LOF and any KNN-reliant algorithms, there are two types of undesirable misclassification. Firstly, a point close to a dense cluster might be categorized as an outlier. Secondly, a distant point might be categorised as an inlier in a vast cluster. With higher dimensions, the accuracy is decreased. The risk of these problems occurring can be reduced by choosing the most optimal k. The strength of the LOF algorithm is that it takes both local and global properties of datasets into consideration. It can perform well even in datasets where abnormal samples have different underlying densities [22]. The LOF process is illustrated in figure 8

4.7 Markov Chain

Markov chain is *memory-less*, which means it is not dependent on previous states, only the next, most-possible state. Using a feature matrix to train the Markov model, we generate a transition matrix. The transition matrix tells us the probability of going from a given stage to another. The transitions with the highest probabilities come from patterns found during the training. This reflects normal traffic, and vice versa for transitions with low probabilities. The hidden Markov model (HMM) is an implementation of the Markov chain and is especially interesting for anomaly detection [11]. Since the class is binomial, there are two states in this HMM. The 29 features are all part of the hidden layer. During the training, the most common patterns of these features are classified as 'normal',

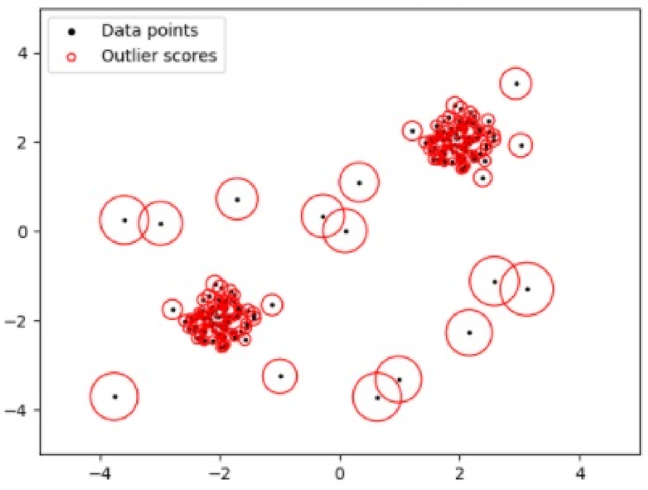


Figure 8: Illustration of Local outlier Factor [22]

where the opposite combinations are labelled as *abnormal*. Since there are only two states, the transition matrix is of the size 2 x 2. Also, an HMM has observation likelihoods, or emission probabilities, which are the probabilities going from the hidden states to the main states [9]. In this project, both of the HMM algorithms use Gaussian emissions, which means the emission probabilities are continuous (Gaussian distribution).

5 EXPERIMENTAL RESULTS

Our evaluation of different techniques are categorized as the following:

5.1 Supervised Techniques:

In our first attempt at implementing an appropriate model for the anomalies dtection, we trained a simple *Random Forest* classifier from *scikit-learn* library with default parameters without any parameter tuning on the imbalanced dataset. The results of the classification report show the recall score of 0.73, f1-score of 0.82, the precision of 0.94 and ROC-AUC of 93.5% . The confusion matrix and ROC curve is shown in figure 9 and figure 10.

Machine learning classifiers such as Random Forest fail to cope with highly skewed training datasets as they are sensitive to the proportions of the different classes and we often expect these algorithms to favour the class with the largest proportion of observations. In the aforementioned case, although the ROC-AUC metric is 93.5%, we see that the f1-score and recall are low. So in the next step, we use different techniques for balancing the dataset to see the effect on Random Forest classification results. By using the *Random Under-sampling* technique, we observe that although we get an improved recall score, the precision is really poor causing the f1-score to drop to 0.1. We also see that ROC-AUC is 97.6% proving our earlier discussion that ROC score is not a decent metric for classification in the imbalanced dataset. The resulting confusion matrix and ROC curve is shown in figure 11 and figure 12.

The classification results for over-sampling with SMOTE and ADASYN techniques are nearly the same, resulting the recall score

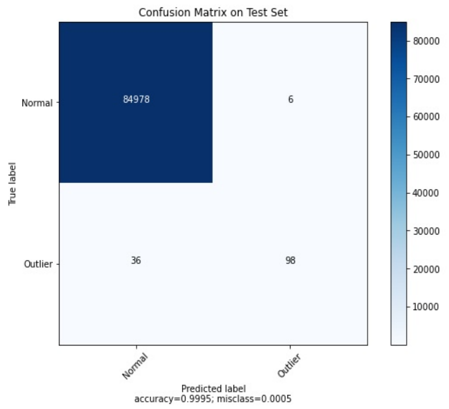


Figure 9: Confusion matrix on Test Set

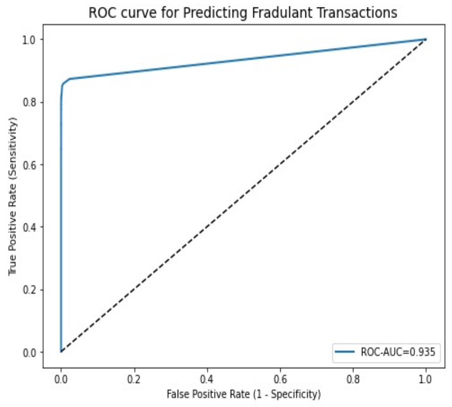


Figure 10: ROC Curve for imbalanced dataset

of 0.79, precision of 0.9 and f1-score 0.84 and ROC-AUC of 97%. The confusion matrix for SMOTE is depicted in the figure 13. We can see that in the case of Random Forest classifier, we could slightly improve the recall and f1-score by making the dataset balanced.

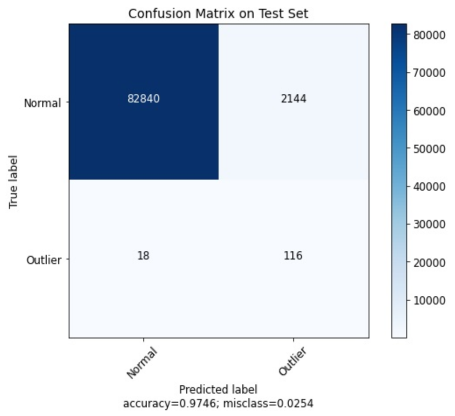


Figure 11: Confusion matrix on Test Set

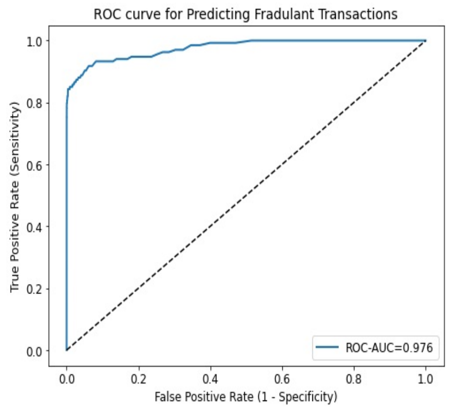


Figure 12: ROC Curve for under-sampled dataset

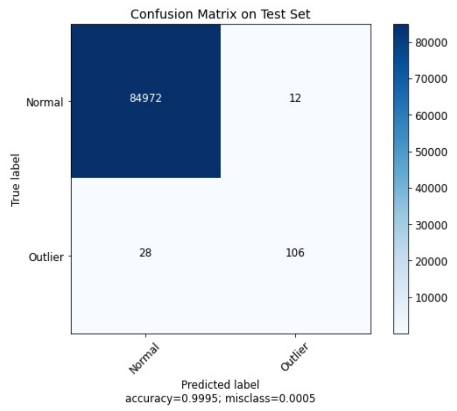


Figure 13: Confusion matrix on Test Set

In next step, we implemented different supervised classification techniques including *Random Forest*, *Naive Bayes Classifier*, *Logistic regression* and *K-Nearest Neighbour* to compare their efficiency while classifying imbalanced, under-sampled and over-sampled datasets. All of the evaluation metrics are calculated and sorted based on f1-score and recall in detecting anomalies in table 1 for comparison of efficiency of each technique. It is worth to mention that the practice of finding best hyper-parameters for tuning the algorithms is done by *GridSearch CV*, a nice feature of scikit-learn library that runs through all the different parameters that is fed into the *parameter grid* and produces the best combination of parameters based on a scoring metric.

From the table 1, it is observed that the best supervised classifier in our study case is Random Forest balanced with SMOTE technique with the highest recall and f1-score. Another finding is that Random Forest is the best supervised algorithm for classification in credit-card dataset. Although over-sampling techniques can improve classification results in multiple cases, under-sampling approaches were unsuccessful in many cases. As per observation, the accuracy and the ROC-AUC is high in all cases, proving that we can not compare the efficiency of the classifier based on these metrics in skewed datasets.

Technique	Accuracy	ROC	PR	Precision	Recall	F1-score
RF SMOTE	0.99	0.97	0.82	0.89	0.79	0.84
KNN imbl	0.99	0.89	0.84	0.94	0.76	0.84
RF ADAS	0.99	0.96	0.80	0.87	0.79	0.83
RF imbl	0.99	0.94	0.81	0.94	0.71	0.81
LR imbl	0.99	0.97	0.71	0.88	0.55	0.67
KNN SMOTE	0.99	0.90	0.72	0.56	0.79	0.66
KNN ADAS	0.99	0.90	0.71	0.56	0.79	0.66
RF Under	0.98	0.97	0.74	0.06	0.86	0.12
LR SMOTE	0.97	0.97	0.73	0.05	0.89	0.10
NB imbl	0.97	0.95	0.39	0.05	0.81	0.10
NB SMOTE	0.97	0.95	0.39	0.05	0.81	0.09
LR Under	0.97	0.97	0.59	0.04	0.88	0.09
KNN Under	0.96	0.95	0.49	0.04	0.88	0.07
NB Under	0.96	0.95	0.41	0.03	0.82	0.07
NB ADAS	0.95	0.96	0.41	0.03	0.84	0.05
LR ADAS	0.91	0.97	0.75	0.01	0.91	0.03

Table 1: Supervised anomaly detection methods

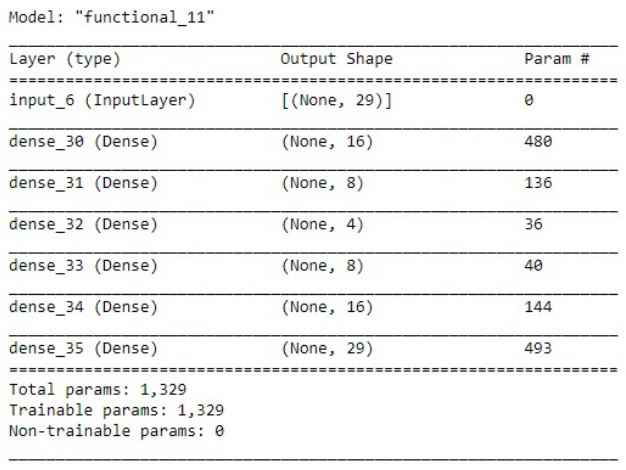
5.2 Unsupervised Techniques

Due to the unavailability of full and accurate labelled datasets in many real-case scenarios, we have to seek an alternate solution to be able to extract the meaningful features from data instances.

5.2.1 ***Auto-Encoders.*** The construction of the Auto-Encoder model in the Keras package is similar to the multi-layer perceptron model. First, we train the model on the normal dataset and then we test it on the whole test set. Although the basic version of the auto-encoder consists of three fully connected layers, the performance of the auto-encoder might be improved by adding new hidden layers. The auto-encoder implemented in this project consists of 7 layers. One input, one output and 5 hidden layers. After pre-processing, the remaining 29 input features were transformed into 16, 8, 4 neurons in the latter hidden layers and then decode into 8, 16 and 29 in the next layers to produce output. The combination of layers is shown in figure 14.

In many use cases of AE for anomaly detection, it is suggested to use *hyperbolic tangent* function for hidden layers, but based on various experiments, we used *RELU* as activation function, because it yields a high level of precision and recall in our case. Furthermore, *mean square error* was set as a loss function parameter, while RMSprop was chosen as an optimizer. After fitting the model and making predictions on the test set, we calculate the reconstruction error between input and output. The results could be summarized in the table 2.

Then in the next step, we need to define a threshold that would separate instances with high reconstruction error from the ones with a lower amount of error. This threshold act as a boundary between normal and fraudulent transactions. After a few trial and errors, we understand that choosing a threshold of 3 would yield the best results. In other words, instances with RE greater than 3 are labelled as anomalies. With auto-encoder, we get a recall of 0.82, precision of 0.43 and an f1-score of 0.56. The outcome of anomaly detection is shown in figure 15. As our second implementation of



```
Model: "functional_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 29)]              0
_________________________________________________________________
dense_30 (Dense)             (None, 16)                480
_________________________________________________________________
dense_31 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_32 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_33 (Dense)             (None, 8)                 40
_________________________________________________________________
dense_34 (Dense)             (None, 16)                144
_________________________________________________________________
dense_35 (Dense)             (None, 29)                493
=================================================================
Total params: 1,329
Trainable params: 1,329
Non-trainable params: 0
_________________________________________________________________
```

Figure 14: Illustration of implemented AE layers

Metric	Reconstruction-Error	True-class
count	4080.0	4080.0
mean	1.296	0.022
std	6.356	0.149
min	0.072	0
25%	0.298	0
50%	0.493	0
75%	0.766	0
max	150.088	1

Table2 : Summary of Statistical measures

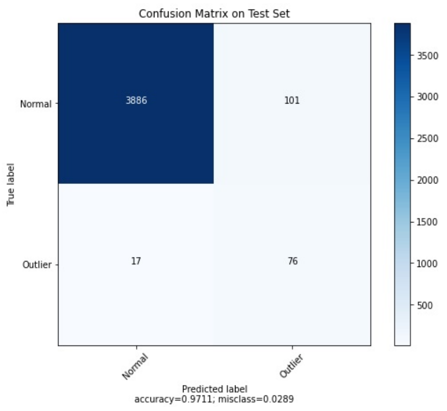


Figure 15: Confusion matrix for auto-encoder

auto-encoders, we added a *sparsity constraint* on the activity of hidden representations to make fewer units "fire" at the given time. This type of auto-encoder, called sparse AE, can be implemented by opting L1 regularization as *activity_regularizer*. We also chose *tanh* as activation function and the rest of the structure is the same as the previous structure. Same as before, after training and making a prediction, we again try to find the best threshold that gives the optimal results. At the threshold of 2.5, we achieve a recall of 0.81, the precision of 0.39 and f1-score of 0.53. By using a sparse auto-encoder, we do not archive better classification results compared to the previous model.

5.2.2 ***Isolation Forest****.* As we discussed comprehensively in the theory behind *Isolation Forrest* algorithm, the path length from the root node to the terminating node, averaged over a forest of random trees, is a measure of normality. Generally, when the algorithm produces shorter path lengths for particular instances, with high probability they are anomalies. The important parameters are *Contamination* which can be interpreted as an estimate of the percentage of data instances considered as an anomaly, *n_estimators* which defines the total number of trees used in the forest and *max_samples* are the maximum number of points each tree should build on. The combination parameter is set to a fraction close to anomaly fraction in the training set and max_samples and n_estimators are chosen 0.5 and 1000 respectively. After training the model on the X_train, we use the decision_function method and plot the average path lengths to identify anomalies depicted in figure 16. By selecting a

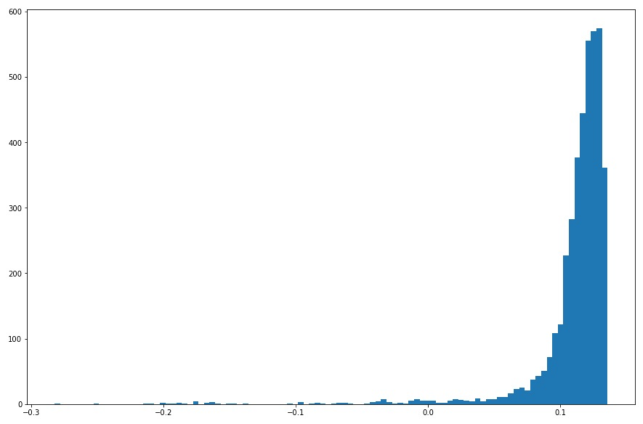


Figure 16: Illustration of average path lengths in Isolation Forrest

threshold path length with trial and error, we see that if we assume that data instances with path length less than 0.03 are anomalies, we can achieve a precision of 0.52, recall of 0.64 and an f1-score of 0.58. The resulting confusion matrix is depicted in figure 17.

5.2.3 ***One-Class SVM****.* According to scikit learn official documents, *One-Class SVM* method is known to be sensitive to outliers and does not perform very well for anomaly detection. That, being said, anomaly detection with One-Class SVM may still being used but requires fine-tuning of hyper-parameter *nu* and *gamma*. to handle outliers. The parameter nu corresponds to the probability of finding instances outside of the frontier. we have chosen this parameter close to the contamination rate of the training set. Moreover, there are different kernel types available for this model including *linear,polynomial*, *sigmoid* and *rbf*. the results of testing different kernels indicated that Radial Basis Function (rbf) would yield the optimal results for this case. *gamma* parameter, which is the kernel coefficient for "rbf", is another parameter needed to be

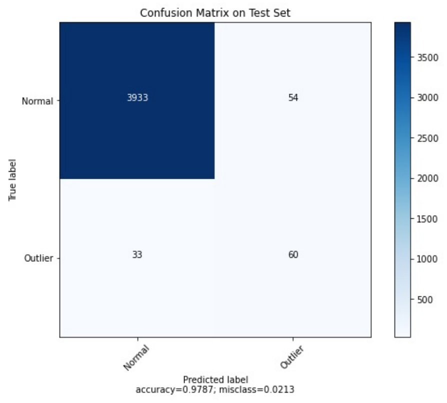


Figure 17: Confusion matrix for Isolation forest

tunned. Experimentation of different amounts for gamma shows that the best value that results in the optimal precision and recall is 0.04. As we expected, One-Class SVM does a relatively poor job in our case of anomaly detection with a recall score of 0.68, the precision of 0.244 and f1-score of 0.361. The results are shown in figure 18.

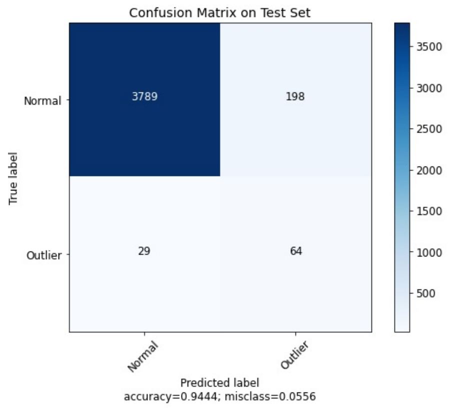


Figure 18: Confusion matrix for One-Class SVM

5.2.4 ***Elliptic Envelope.*** The basis of this algorithm is that we try to define the "distribution" and “shape” of the data instances and consequently define anomalous observations as instances that stand far enough from the fit shape. By implementation of scikit-learn *Elliptic Envelope*, we try to fit an ellipse to the central data points, ignoring points outside this central mode. again the *contamination* parameter in this algorithm is the amount of contamination of the data set. This algorithm gives us results as a recall of 0.65, a precision of 0.71 and an f1-score of 0.69.

5.2.5 ***K-means Clustering.*** To implement this algorithm, we use *MiniBatchKMeans* from scikit-learn. In the beginning, we define the number of clusters randomly as 20. We then obtain and store cluster centroids in the memory. The distance of each point in the given dataset is calculated with respect to their cluster centroids. We have also defined a cut off point to define what is an outlier

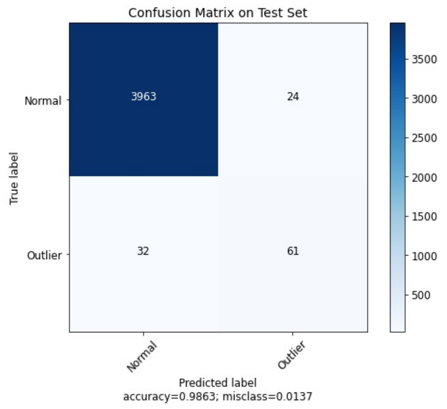


Figure 19: Confusion matrix for Elliptic Envelope

and what is not i.e., everything with a distance larger than the top 95th percentile should be an outlier. Now, we use an elbow curve to determine the right number of clusters that we need to use. From the figure 20, we get that the right number of clusters should be 2. The anomaly detection in this method has a recall score of 0.78, a precision of 0.27 and an f1-score of 0.40 (figure 21).

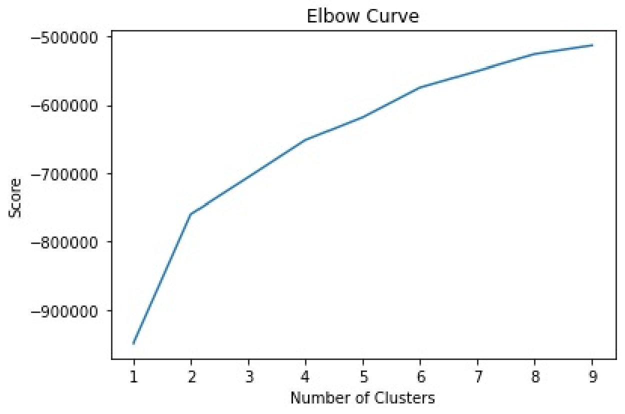


Figure 20: Illustration of elbow curve

5.2.6 ***Markov Chain.*** A Markov chain consists of several parameters, number of states and transition probability matrix, which contain the probability of every possible transition. With the libraries, *GaussianHMM* and *GMMHMM* both can calculate the latter, and need only to have specified the former. In our case, there are only two possible states, normal and fraudulent transactions. The results of the GaussianHMM method was pretty bad in the detection of anomalies and for the GMMHMM case, Although the recall score is 0.9, the precision and f1-score are low around 0.01. In another word, this algorithm can detect anomalies with too many false positives(figure 22)

5.2.7 ***Local Outlier Factor (LOF).*** The most important parameter with LOF is the k with KNN. A lot of scikit-learn libraries have their own best_parameters_object call, but Local Outlier Factor does not. Therefore, it was important to find an alternative. we

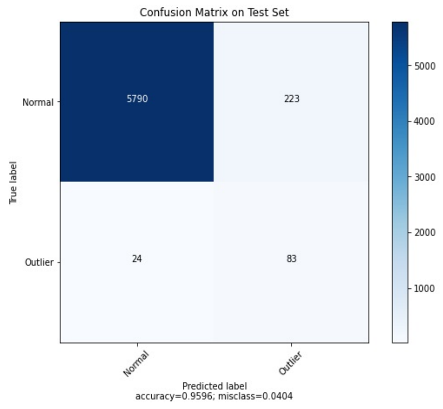


Figure 21: Confusion matrix for k-means (k=2)

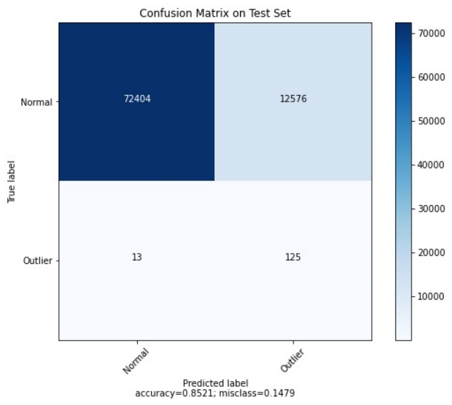


Figure 22: Confusion matrix for Markov Chain (GMMHMM)

compared the metrics with different k-values. Due to long computation time, we only tested k values in a high interval. The interval used was from 5 to 1000 with a step size of 55. The result of the classification with the best value of k can be observed in figure 23. This method yields recall of 0.52, recall of 0.53 and precision of 0.55.

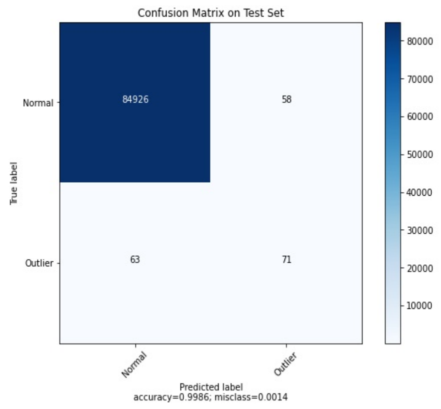


Figure 23: Confusion matrix for LOF

The evaluation metrics for different unsupervised methods are gathered and sorted based on f1-score and recall in detecting anomalies in table 3 for comparison of efficiency of each technique.

Technique	ROC	PR	Precision	Recall	F1-score
Elliptic Envelope	0.82	0.69	0.72	0.65	0.68
Isolation Forest	0.81	0.59	0.52	0.64	0.58
Auto-Encoder	0.89	0.62	0.43	0.82	0.56
Sparse AE	0.88	0.60	0.39	0.81	0.53
LOF	0.88	0.59	0.55	0.53	0.52
K-means	0.86	0.52	0.27	0.78	0.40
One-Class SVM	0.82	0.47	0.24	0.68	0.36
Markov Chain	0.87	0.45	0.01	0.9	0.01

Table 3: Unsupervised anomaly detection methods

6 CONCLUSION

Anomaly(outlier) detection is defined as the practice of finding observations that do not conform to the expected pattern of other data instances. These outliers can cause adverse events such as fraudulent activities in banking systems. Due to huge monetary loss to individuals and banks, fraud detection is one of the most researched areas in anomaly detection.

In this project, various supervised and unsupervised techniques were experimented and their performance measures were compared. By opting for the tuned, appropriate parameters, conventional supervised methods, when training in a balanced mode, can result in accurate and descent classification outcome. The problem with supervised methods is their need for accurate labels which is impractical in many applications. Furthermore, the nature of fraudulent transactions(anomalies) is constantly changing, making it difficult for supervised techniques to find anomalies.

On the other hand, the biggest advantage of unsupervised methods is that they do not need a fully labelled dataset. However, the biggest challenge of the majority of these techniques is the selection of appropriate threshold and hyper-parameter tuning to achieve optimal results. In other words, the expert needs to have a broad knowledge domain about the application of these algorithms. In many scenarios, although we are successful in detecting anomalies, the rate of false positives is high meaning there exist many normal instances that are categorized as an anomaly.

7 ACKNOWLEDGEMENTS

We are especially thankful to our course lecturer "Vinay Jayarama Setty" for allowing us to work on this project. We would also like to thank our teacher assistants Bjarte Botnevik and Eiriksak for their constant help and advice during this project.

8 AUTHORS AND SHARE OF WORK

This project was done by the collaboration of Alireza Hossein Zadeh, Sondre Tennø, Prava Thapa and Håkon Teppan. All tasks were completed through active discussions and joint efforts. However, to name the specific tasks, algorithms and responsibilities of each group member, they are mentioned below:

- **Alireza Hossein Zadeh:** Implementation of supervised techniques (balanced and imbalanced methods), implementation of Auto-encoder
- **Sondre Tennø:** Implementation of supervised techniques(balanced and imbalanced methods), implementation of Isolation Forest
- **Prava Thapa:** Pre-processing and Data visualization, implementation of One-Class SVM, Elliptic Envelope and K-means Clustering
- **Håkon Teppan:** Pre-processing and Data visualization, implementation of Markov chain and Local Outlier Factor

REFERENCES

[1] Mahbubul Alam. 2020. Machine learning for anomaly detection: Elliptic Envelope. https://towardsdatascience.com/machine-learning-for-anomaly-detection-elliptic-envelope-2c90528df0a6

[2] Sridhar Alla and Suman Kalyan Adari. 2019. *Beginning anomaly detection using python-based deep learning*. Springer.

[3] Will Badr. 2019. Auto-Encoder: What Is It? And What Is It Used For? (Part 1). https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

[4] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.

[5] Jason Brownlee. 2021. Random Oversampling and Undersampling for Imbalanced Classification. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification

[6] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.

[8] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[9] Sean R Eddy. 2004. What is a hidden Markov model? *Nature biotechnology* 22, 10 (2004), 1315–1316.

[10] Michael J. Garbade. 2018. Understanding K-means Clustering in Machine Learning. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[11] Abida Haque, Alexandra DeLucia, and Elisabeth Baseman. 2017. Markov chain modeling for anomaly detection in high performance computing system logs. In *Proceedings of the Fourth International Workshop on HPC User Support Tools*. 1–8.

[12] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 1322–1328.

[13] Maryamsadat Hejazi and Yashwant Prasad Singh. 2013. One-class support vector machines approach to anomaly detection. *Applied Artificial Intelligence* 27, 5 (2013), 351–366.

[14] The imbalanced-learn developers. 2021. ADASYN. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html

[15] The imbalanced-learn developers. 2021. imbalanced-learn documentation. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html

[16] The imbalanced-learn developers. 2021. SMOTE. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[17] Applied Research in Cloudera Fast Forward. 2020. Deep Learning for Anomaly Detection. https://ff12.fastforwardlabs.com/#how-to-decide-on-a-modeling-approach%3F

[18] Eryk Lewinson. 2018. Outlier Detection with Isolation Forest. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

[19] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[20] Mahdi Rezapour. 2019. Anomaly detection using unsupervised methods: credit card fraud case study. *Int J Adv Comput Sci Appl* 10, 11 (2019).

[21] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. 1999. Support vector method for novelty detection.. In *NIPS*, Vol. 12. Citeseer, 582–588.

[22] scikit-learn official docs. 2018. Novelty and Outlier Detection. https://scikit-learn.org/stable/modules/outlier_detection.html#outlier-detection

[23] sk-learn official docs. 2018. IsolationForest. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

[24] Roy Thomas and JE Judith. 2020. A Novel Ensemble Method for Detecting Outliers in Categorical Data. *International Journal* 9, 4 (2020).

[25] Kushal Vala. 2021. ADASYN: Adaptive Synthetic Sampling Method for Imbalanced Data. https://towardsdatascience.com/adasyn-adaptive-synthetic-sampling-method-for-imbalanced-data-602a3673ba16

[26] the free encyclopedia Wikipedia. 2020. Oversampling and undersampling in data analysis. https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis

[27] Syoya Zhou. 2018. What happens in Sparse Autoencoder. https://medium.com/@syoya/what-happens-in-sparse-autencoder-b9a5a69da5c6