Credit Card Fraud Detection Using Supervised Machine Learning Algorithms – Experiments with Resampling, Feature Selection, and Ensemble Models

Adam Toth
210505924
Ms Zunaira Nadeem
MSc FT Computer Science
(Conversion)

***Abstract*— Advances in e-commerce systems, and progresses made in the communication and payment technologies have accelerated the number of credit card transactions both online and offline purchases of goods and services. Simultaneously, however, the increased use of credit cards made frauds associated with such transactions more pervasive, costing financial institutions billions of dollars annually.**

Considering the major challenges already identified by previous works regarding credit card fraud (CCF) detection, such as class imbalance of datasets; improving performance by various feature selection (FS) techniques; and that no single algorithm is a perfect fit to a particular dataset, the objective of this paper is manifold.

The paper first analyses five supervised machine learning algorithms commonly used in fraud detection on a well-known, 'raw' real-world dataset and evaluates their performance. It then applies several resampling and FS techniques to combat the class imbalance problem and to enhance performance of the algorithms. Finally, the best performing classifier is selected, and paired up with classifiers, creating ensemble models to examine whether their combined workings improve their capabilities in detecting fraud cases.

The overall aim of this experiment, therefore, is not only to find the most efficient combination of the above techniques and classifiers and see whether they can outperform existing results, but also to highlight the incremental improvements in performance by each consecutive steps – resampling, feature selection, and ensemble models.

Keywords—resampling, feature selection, ensemble models, credit fraud, machine learning

1. I . INTRODUCTION

Recent advancements in various technologies and the exponential growth of the Internet have made the rapid expansion of online and offline payments made using credit cards available. The already employed security measures to protect such transactions against fraudsters (e.g., data encryption, tokenization, fraud detection systems) are constantly challenged by criminals with a detrimental financial impact that cannot be underestimated, and which quantifies the significance of the issue. According to the Nilson Report (2021), global card payment fraud loss amounted to $32.20 billion in 2021 and is projected to increase to $49.32 billion by 2030.

Whilst detection and prevention of such frauds has shifted from manual techniques – which are impractical in the age of big data – towards sophisticated, data-driven machine learning algorithms, there are multiple challenges associated with the latter. Thennakoon et al. (2019) summarized the major problems as follows.

First, there is a general lack of real-world datasets due to privacy concerns. Even if a dataset is available, it might no longer be accurate, since fraudulent behavior and its patterns are dynamically evolving with time (Randhawa et al., 2018).

Second, CFF datasets' class distribution is highly imbalanced, since only a small fraction of the total transactions is fraudulent. This raises further challenges as to find an optimal sampling approach, so the selected classifiers do not undermine the importance of the minority class; and appropriate evaluation metrics that consider the skewed distribution of class instances (Pozzolo et al., 2014).

Third, extracting the most important features from a high feature dimension space with various selection techniques will improve not only the final accuracy but overall interpretability of the models too.

Finally, improved cross-validation techniques and algorithm selection (either standalone or ensemble methods) can also improve the performance of a proposed CCF detection architecture by combining multiple classifiers, resulting in reduced generalization error of predictions (Osamor and Okezie, 2021).

This paper proposes a four-stage process for producing a final ensemble model which is equipped with the best-performing combination of resampling and feature selection techniques through experimentation, giving the highest performance. The first stage analyses five supervised machine learning algorithms (with default parameters) on a raw dataset, evaluated using various performance evaluation metrics. All supervised ML algorithms assume the availability of labels of past transactions (fraud, or non-

fraud). The five algorithms are Logistic Regression (LR), K-Nearest Neighbours (KNN), Decision Tree (DT), Random Forest (RF), and Naïve Bayes (NB).

The second stage uses the same five algorithms, but experiments with three under- and oversampling methods respectively, to see how performance of the classifiers change.

The third stage repeats the process of the second stage, but it is further analyzed with one filter and wrapper feature selection technique, respectively. The best model is then selected and carried over to the fourth stage.

At the fourth stage, the abovementioned algorithm has a guaranteed space to be combined with several other algorithms, creating various ensemble models.

The remainder of this paper is structured as follows: Section II presents a literature review of similar work. Section III introduces the dataset being used, the resampling and feature selection techniques chosen, gives a short description of the selected classifiers, and depicts the overview of the proposed experimental architecture. Section IV examines the evaluation metrics used for performance analysis. Section V includes the results and discussion, and future work possibilities.

II . Related Work

The European Credit Card Fraud Detection dataset (ECCFD) is one of the most used datasets in CCF research. The following papers all utilised it with a varying number of techniques – at least partially – making their results an ideal benchmark for this paper. Khatri et al. (2020) compared DT, KNN, LR, RF, and NB on raw data, without any data preparation. The best performing Naïve Bayes and KNN achieved recall values of 85.15% and 81.19%, respectively, whilst in terms of precision, RF and KNN scored 93.83% and 91.11%.

Alfaiz and Fati (2022) proposed a two-stage architecture to analyse LR, KNN, DT, NB, RF, GBM, LightGBM, XGBoost, and CatBoost. The classifiers were compared on raw data in the first stage, after which the best three algorithms were passed to the second stage, where 11 undersampling, 6 oversampling, and 2 combinations of under- and over-sampling techniques were analysed with respect to the three classifiers to find the best performing combinations. RF and KNN scored precision values of 94.87% and 83.75%, and recall scores of 78.46% and 9.56%, respectively (both behind CatBoost and XGBoost). Regarding AUC and F1-Score, RF was in the best three algorithms (97.42% and 85.88%), whereas NB achieved the lowest values of AUC and Recall (57.36% and 14.78%). In the second stage, RF combined with undersampling techniques RENN and IHT achieved best performance of 78.86% recall value (95.82% precision), and 99.59% recall (0.0018% precision), respectively.

Mishra and Ghorpade (2018) analysed LR, SVM, RF, and Gradient Boosted Trees (GBT) with Random Undersampling applied, in addition to their hyperparameters optimised. RF, and LR achieved 95.7% and 90.78% precision, and recall scores of 92.5% and 93.87% on undersampled data, respectively. The paper also created an ensemble model, consisting of LR, SVM and GBT with majority voting, which achieved better performance than all standalone algorithms, namely 94.4% for precision and 91.8% recall. Further papers with only SMOTE applied on datasets can be found in Dornadula and Geetha (2019), Puh and Brkic (2019), and Kumar et al. (2019).

Saheed et al. (2020) applied Genetic Algorithm for feature selection of the German credit dataset (Hofmann, 1994) without any resampling, to improve fraud detection of NB, RF, and SVM. The paper categorized attributes into first and second priority features, according to their importance. With the former set, NB and RF achieved 94.3% and 96.4% recall and 94.7% and 96.5% precision scores, respectively. Contrarily, with the latter set, both classifiers performed considerably worse.

Similarly, Ileberi et al. (2022), in conjunction with SMOTE resampling, applied GA with RF as its fitness function on the ECCFD. RF, DT, ANN, NB, and LR's performances were analysed. After five optimal feature vectors were generated by GA, GA-RF overall optimal accuracy of 99.98% (precision 95.34%, recall 77.87%), and GA-DT accuracy of 99.92% (precision and recall of 75.22%) were achieved, among other remarkable results. The paper also showcased that utilizing all features of the dataset or randomly selecting them will result in seriously underperforming classifiers, compared to GA-led feature selection.

Esra et al. (2012) experimented with six filter-based feature selection techniques (Information Gain, Gain Ratio, Symmetrical Uncertainty, Relief-F, One-R, Chi-square) before applying the NB, MLP, and DT-J48 algorithms on fifteen different, real-world datasets. Up to 15.55% improvement in classification accuracy were observed, however, each three classifiers seemed to improve by and be sensitive to different feature selection techniques.

Varmedja et al. (2019) analysed LR, RF, NB, and MLP after applying SMOTE resampling. For FS, a feature selector tool by Will Koehrsen was applied, by which 3 attributes that did not contribute to the cumulative importance of 95% were removed, leaving 27 features for additional experimentation. LR (recall: 91.84%, precision: 58.82%), NB (recall: 82.65%, precision: 16.17%), RF (recall: 81.63%, precision: 96.38%), MLP (recall: 81.63%, precision: 79.21%) performance metrics were achieved.

Malik et al. (2022) selected LR, RF, DT, XGBoost, SVM, NB, AdaBoost and LGBM for analysis (all with default parameters) on a highly imbalanced (3.5%) dataset containing about half a million credit card transactions, with 432 input features (both numeric and categorical). To rebalance the dataset, a combination of undersampling (Edited Nearest Neighbours) and oversampling (SMOTE) was applied. Regarding feature selection, the paper considered that filter FS approaches look for features as individuals, thereby features dependent on each other will be missed by this approach when combined with other features; furthermore, it considers the disadvantages of wrapper techniques, that is, their high computational cost, or their dependence on algorithms calculating the fitness function of features. Hence, a hybrid FS approach is proposed,

Authors	Hyperparameters optimised	RSMP	FS	ENSB	LR		DT		RF		NB		KNN		ENSB	
					REC	PRC	REC	PRC	REC	PRC	REC	PRC	REC	PRC	REC	PRC
Khatri et al. (2020	-	-	-	-	63.34	87.67	79.21	85.11	75.25	93.83	85.15	6.56	81.19	91.11		
Alfaiz et al. (2022)	-	+	-	-					78.86 99.59	95.82 0.0018						
Mishra A., Ghorpade C. (2018)	+	+	-	+	93.87	90.78			92.5	95.7					91.8	94.4
Ileberi et al. (2022)-	-	-	+	-	46.9	34.64	72.56	65.07	72.56	95.34	57.52	15.85				
Varmedja et al. (2019)	-	+	+	-	91.84	58.82			81.63	96.38	82.65	16.17				

Table 1. Literature Review of Results (RSMP = Resampling, FS = Feature Selection, ENSB = Ensemble Model)

integrating both filter and wrapper approaches. A correlation-based filter first removed highly dependent features (correlation higher than 0.8) that would not be of use to the prediction model but just promote overfitting, after which a wrapper technique, SVM-Recursive Feature Elimination was applied. Most algorithms' (except NB) AUROC scores ranged between 0.66-0.71. Best performer of this stage was AdaBoost, hence it was selected as the single baseline model to be combined with the rest of the algorithms to create the best hybrid model, which was chosen to be AdaBoost + LGBM (with recall: 64%, precision: 97%). Decreasing performance of LR and SVM once combined with AdaBoost proved that ensemble machine learning models are not always a guarantee for higher performance.

Finally, Dhankhad et al. (2018) examined LR, RF, KNN, SVM and a final super classifier (ensemble) model, after SMOTE resampling. The super classifier outperformed all single algorithms in most evaluation metrics, whilst Random Forest came second.

III . RESEARCH METHODOLOGY

Dataset

The real-world dataset being used in this paper was obtained from Kaggle (2021). It contains 284,807 transactions made by credit cards over two days in September 2013 by European cardholders, out of which only 492 are frauds (0.172%), making it a highly unbalanced dataset. It includes thirty numerical input features, consisting of the PCA transformed 'V1' to 'V28' (transformed due to confidentiality issues); 'Time', showing the seconds elapsed between the first and all following other transactions; and 'Amount' representing the transaction amounts. Finally, the binary response variable 'Class', where 1 represents fraudulent transactions, and 0 otherwise. There are no missing values or duplicates in the dataset.

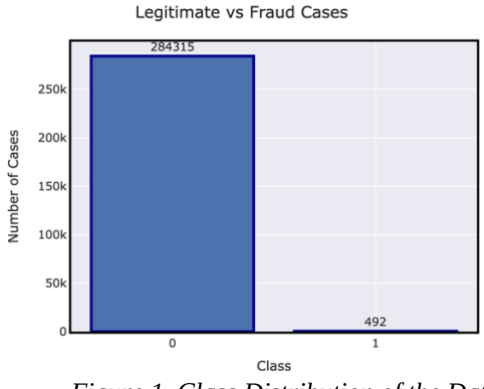

Figure 1. Class Distribution of the Dataset

Resampling

Since most machine learning algorithms assume equal distribution of classes in a dataset, the class imbalance problem present must be corrected for accurate predictive modelling performance and interpretability.

Undersampling produces a balanced training dataset by preserving all minority class instances, whilst decreasing the size of the majority class examples. Contrarily, oversampling keeps the majority class instances, whilst replicating the minority class instances until balance of classes are achieved (Lucas and Jurgovsky, 2020).

This paper experiments with Random Undersampling (RUS), Near Miss (NearMiss), Tomek Links (TomekLinks) under-sampling techniques, and Random Oversampling (ROS), Synthetic Minority Oversampling (SMOTE), and Adaptive Synthetic (ADASYN) Oversampling methods. Each selected resampling techniques are briefly described in the followings.

During RUS, all data points of the minority class are preserved, whilst instances from the majority class are randomly removed from the training set until balance between the classes is achieved. A disadvantage of this approach is that some useful information of the majority class can be lost, however one of its major advantages is fast execution (Dubey et al., 2014).

The NearMiss-1 method used in this paper selects instances of the majority class based on the smallest averaged distance from the N closest (default=3) minority class data points, and removes them (Mani and Zhang, 2003).

The Tomek Links method selects majority class instances to be removed if they satisfy the following rule:

$$dist(a, b) < dist(a, c) \text{ and } dist(a, b) < dist(b, c),$$

where dist() is the distance between two samples of different class. That is, there said to be a Tomek's link between such two samples if they are the nearest neighbours of each other. The default sampling strategy, which was used in later experiments is 'auto', meaning only majority sample Tomek Links were removed during undersampling (Swana et al., 2022).

When Random Oversampling is performed, whilst all majority samples of the dataset are retained, minority data instances are randomly picked and duplicated with replacement, until the desired class balance is achieved in the training set. Since the same minority class examples are duplicated many times, algorithms performing on the resulting training set can be prone to overfitting, as no new information is provided by the duplicates, leading to a loss of generalization on the test set.

During SMOTE, minority class instances are oversampled (whilst all majority class cases are kept) by creating synthetic samples in the following way. SMOTE first randomly selects a minority class sample 'a' and determines its K-Nearest Neighbours based on the Euclidean distance between them. The line segment between the feature vector 'a' and its nearest neighbour 'b' is then multiplied by a random number between 0 and 1 and added to the feature vector being considered, creating a random, new synthetic point in the feature space (Chawla et al., 2002). Whilst this technique avoids ROS's possible overfitting caused by the exact replication of instances, or information loss of RUS, SMOTE does not take it into account that K-nearest neighbours of minority class instances can be from the majority class, thereby increasing overlap of classes and, resultingly, additional noise in the training dataset.

The general idea behind ADASYN is the use of weighted distribution for different minority class instances according to their difficulty level of learning – for harder to learn instances more data is generated than those samples that are easier to learn (He et al., 2008). The algorithm works as follows. As a first step, the algorithm calculates the number of synthetic minority data instances to be generated, G:

$$G = (m_{maj} - m_{min})\beta \qquad (1)$$

where m_{min} and m_{maj} are the total number of minority and majority class instances in the dataset, and β is the desired class ratio to be achieved (in this paper, 1).

At the second step, ADASYN finds all minority samples in the data, and identifies their k nearest neighbours (default=5), thereby creating separate neighbourhoods for all minority class examples. The dominance of the majority class is calculated in each neighbourhoods using:

$$r_i = \frac{\#\ of\ majority\ class\ instances}{k} \qquad (2)$$

Higher r_i will indicate neighbourhoods with heavier majority class dominance, and that such neighbourhoods are harder to learn. During the third step, all r_i is normalised, hence they become density distributions equalling to 1:

$$\hat{r_i} = \frac{r_i}{\sum r_i} \quad \rightarrow \quad \sum \hat{r_i} = 1$$

Using this calculated density distribution, the number of synthetic samples to be generated is calculated using

$$G_i = G\hat{r_i}$$

As more instances are generated in harder-to-learn neighbourhoods, gives ADASYN its adaptive nature implied by its name. At the final step, G_i data is generated in all neighbourhoods by

$s_i = x_i + (x_{zi} - x_i) \curlywedge$,

where $\curlywedge$ is a random number between 0 and 1, s_i is the generated synthetic instance, and x_i and x_{zi} are two minority examples from the same neighbourhood. That is, from the k nearest neighbours of data x_i, ADASYN randomly selects a minority sample x_{zi}, and through their linear combination, a new synthetic example is generated (He et al., 2008).

Despite its adaptability, synthetic data samples generated in neighbourhoods with dominance of the majority class may resemble the majority class too closely, resulting in increased false positives of classifiers.

Feature Selection

Feature selection (FS) is a crucial data preparation step, involving the omission of redundant or irrelevant features of high-dimensional datasets, whilst preserving relevant information that provide good prediction results (Thudumu et al., 2020). A successful FS' advantages include decreased training time of machine learning models; lowered complexity, which enhances interpretability; improved performance metrics; enhanced generalization of the model by less overfitting; and finally, the avoidance of the curse of dimensionality (Chandrashekar and Sahin, 2014).

Two common approaches of FS are filter- and wrapper-based methods. The former analyses each features independent from the classifiers, after which they are ranked by importance. That is, a subset of features is selected based on their relationship with the target and then fed into the classifiers. Contrarily, the latter iteratively takes a subset of the available attributes, analyses a classifier's performance on each, and then it selects the subset on which the classifier performed with the highest performance (Chandrashekar and Sahin, 2014).

This paper utilizes ANOVA-F test as filter method by which features with the highest F-scores are selected (number of selected features is set to 20). The chosen wrapper method is Recursive Feature Elimination (RFE), which works as follows. RFE wraps a Decision Tree Classifier at its core, which is fit to the training dataset, computing and ranking the importance of each feature. Once feature importance is determined, it then iteratively removes the least important features until a predefined size of feature subset is achieved (also set to 20) (Chen and Jeong, 2007).

Ensemble Models

This paper's final stage centres around Majority Voting-based ensemble models (EMs). The idea of EMs is to use multiple machine learning base models acting as a single one in order to achieve better predictive performance and generalization than its individual building blocks. Majority Voting (MV) refers to the method(s) through which the individual classifiers act as one – this analysis implements two variations of MV, Simple Majority Voting (SMV) and Weighted Majority Voting (WMV). At its core, the former will make separate class predictions for each data instance, but the final, united class label is given to examples that were predicted by the majority of the classifiers. Formally expressed,

$$c = S(x) = argmax T_k \qquad (1)$$
$$k \in \{1, \dots, Q\}$$

where c is the final predicted class of data instance x, S is a set of algorithms unified in the ensemble ($S = \{A_1, A_2, ..., A_M\}$), and T_k is the total count of same class ($Q = \{0, 1\}$ in this paper) votes placed on an instance.

WMV works the same way, but it recognizes that certain classifiers might perform better than others in the algorithm set S. It allows higher weights to be given to more skilful algorithms, thereby further improving their unified performance. The total count of class votes, T_k, is modified to

$$T_k = \sum_{l=1}^{M} w_l(A_l)\, x\, F_k(c_l) \qquad (2)$$

in order to recognize weight differences (where $F_k(c_l)$ is classifier-specific counting function). Ultimately, a class label that receives the highest total weight is chosen as prediction (equations based on Bouziane et al. 2011). A variation of WMV, weighted 'soft voting' is also experimented with in this paper, in which rather than taking the mode of predicted labels of classifiers as vote, their probability output is weighted and summed up. The class label with the highest sum of weighted probabilities is promoted as final decision.

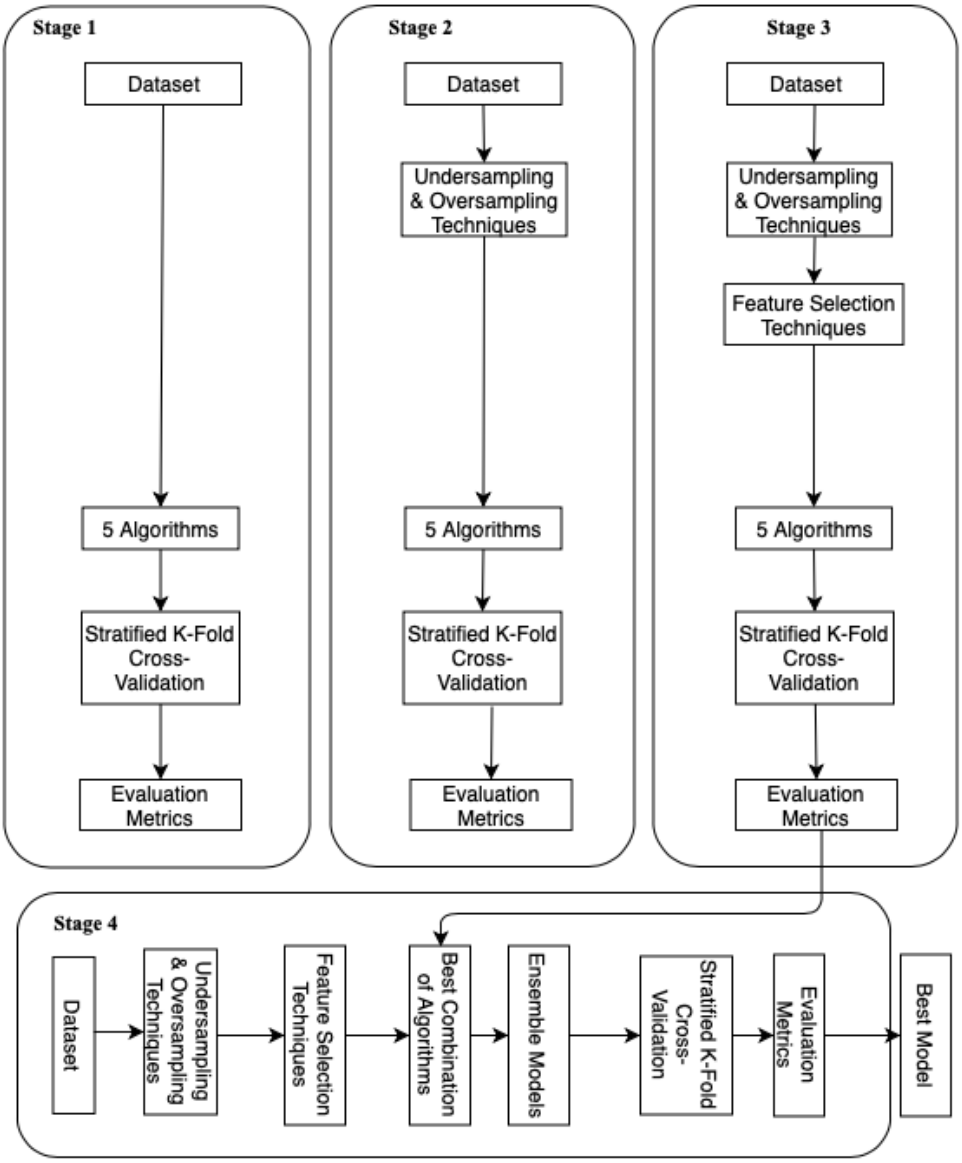

Figure 2. Proposed Architecture for Experiments

Classifiers

There are five machine learning algorithms that are being used in the proposed stages. They are LR, DT, RF, NB, and KNN. Each classifiers' parameters are set to default, except KNN, where "n_neighbours=3", and LR, where "solver='lbfgs'".

Logistic regression uses maximum likelihood estimation to model the probability of samples belonging to different binary classes.

Decision Trees partition a dataset's feature space by using a set of binary splitting rules (e.g., 'true' or 'false') , resulting in an inverted tree-like structure. The partitioning continues until a specified stopping criteria is satisfied. After partitioning, predictions are given based on the class label that has a majority representation (Song and Lu, 2015).

The Random Forest algorithm is an ensemble model in itself, consisting of multiple decision trees. The algorithm first creates a bootstrapped dataset from the training data (with replacement), after which a wide variety of decision trees are built on randomly selected subsets of variables of that dataset. When making predictions, new data samples are run through the abovementioned decision trees, and the majority class label given to an example across all ensembles trees are then the final prediction of the algorithm (Pal, 2007).

K-Nearest Neighbour searches through the entire dataset and analyses each instance based on their k-nearest neighbours using some distance measures (Minkowski distance by default). When KNN is used for classification, class prediction of a data point is given by the most frequently occurring class label in the k-nearest instances (k=3 in this paper).

Naïve Bayes is a probabilistic classifier with a strong assumption of the features of a dataset being independent. It leverages Bayes' Theorem, from which the priori and posterior probability of classes are calculated. A data instance is assigned to a class label with the highest conditional probability (Devi and Sumanjani, 2015).

Experimental Configuration

This paper's experiments were carried out on Google Colab. The computer hardware specifications are as follows: MacBook 2.3 GHz Dual-Core Intel Core i5 (4 Cores). Scikit-Learn's Machine Learning framework was utilised for this research.

IV . EVALUATION

To evaluate the performance of the proposed CCF detection approach, stratified 5-fold cross-validation method is applied. That is, the dataset is randomly split into 5 unique subsets of equal size, out of which 4 subsets (80%) are used to train the models, whilst the remaining 1 subset (20%) is used for testing and providing the evaluation metrics. The process is repeated five times until each different 20% subset have been used as a test set. The average performances of the test sets are calculated, giving the final, total performance of the utilised algorithms. To avoid data leakage, SkLearn pipelines are used for both resampling and FS techniques, meaning these techniques are only applied on the training folds, whilst the test folds contain the original class distribution and features.

This paper considers evaluation metrics Precision, Recall, Accuracy, and F1-Score. Each of these metrics can be calculated from a model's confusion matrix, which presents the difference between the model's prediction and ground truth labels of the dataset. It defines the following four categories:

1. TP (true positive) – number of fraudulent transactions correctly classified.
2. FP (false positive) – number of legitimate transactions classified as fraud.
3. FN (false negative) – number of fraudulent transactions classified as legitimate.
4. TN (true negative) – number of legitimate transactions correctly classified.

The abovementioned metrics are calculated as follows:

5. $Precision = \frac{TP}{TP+FP}$

6. $Recall = \frac{TP}{TP+FN}$

7. $Accuracy = \frac{TP+TN}{TP+TN+TP+FN}$

8. $F1 - Score = 2\frac{RC\ x\ PR}{RC+PR}$

Additionally, Area Under the Curve (AUC) value is calculated, measuring the area under the Receiver Operating Characteristic Curve (ROC), with a value range of 0 to 1 (higher the score, the more efficient the model is).

This paper's experiments primarily consider Recall and Precision values (in this order) for the following reasons. First, classification accuracy of algorithms on heavily imbalanced datasets are misleading and unreliable, as very high accuracy scores can be achieved simply by predicting the majority class of the dataset, also known as the 'accuracy paradox' (Galar et al., 2012).

Second, for selecting between Precision and Recall, one must consider their economic costs and benefits in a given use case. Whilst Precision is advantageous to use when the cost of false positives is high, Recall is preferred when high costs are associated with false negatives. High number of false positives (low Precision) may involve increased cost of human labour for analysing flagged transactions, or the inconvenience caused by temporarily blocked credit cards. However, high number of false negatives (low Recall) will result in reimbursements costs or the increased cost of insurance premiums protecting financial institutions against uncaught fraudulent transactions. Ideally, a model would both have a high Recall and Precision score.

Third, whilst the F1-Score is a function of both Precision and Recall (their harmonic mean), its default configuration gives equal weight to both metrics, which is undesirable for the abovementioned reasons. Nonetheless, all mentioned metric results for each stages of the experiment can be found in the Appendix.

V. RESULTS AND DISCUSSION

Stage 1

As expected, all metrics obtained at the first stage of the proposed architecture are in close range of those observed in literature. Best performing Random Forest only slightly outperformed Decision Tree in terms of Recall (77.04 and 77.03, respectively), however, the former's Precision outshines all other four classifiers' metrics (95.26). KNN seems to be the least able to correctly identify fraudulent transactions, with Recall score as low as 8.94.

	Recall	Precision
LR	0.62	0.8451
DT	0.7603	0.7534
RF	0.7704	0.9526
NB	0.6462	0.1467
KNN	0.0894	0.8413

Table 1. Raw Data Results

Stage 2

Considering the imbalanced nature of the dataset, it is not surprising that at the second stage, all classifiers achieved better results at least with one undersampling or oversampling method than the benchmark metrics obtained on raw data. The following results are the best combination of classifiers and resampling techniques. LR-TomekLinks achieved Recall of 62.98 (1.58% increase), and Precision of 84.65 (0.16% increase). DT-TomekLinks attained Recall of 76.41% (0.5% increase), and Precision of 76.69 (1.79% increase). RF-TomekLinks scored Recall of 78.12 (1.4% increase), and Precision of 95.48 (0.23% increase). NB-SMOTE combination resulted in Recall of 75.82 (17.3% increase), and Precision of 15.27 (4.08% increase). Finally, KNN-ROS achieved Recall of 21.24 (138% increase), however its precision decreased by 68.76% to 26.28. Overall, RF, DT and LR still confidently outperforms the other

classifiers. Although NB's recall significantly improved, and now is in par with RF and DT, its Precision still lags behind.

	Recall	Recall incr.	Precision	Precision incr.
LR - TomekLinks	62.98	1.58%	0.8465	0.16%
DT - TomekLinks	76.41	0.50%	0.7669	1.79%
RF - TomekLinks	78.12	1.40%	0.9548	0.23%
NB - SMOTE	75.82	17.30%	0.1527	4.08%
KNN - ROS	21.24	138%	0.2628	-69%

Table 2. Resampled Data Results

Stage 3

Similar to the previous stage, further performance improvements were achieved either by coupling the classifiers with FS techniques alone, or with a combination of resampling and FS techniques. The following performance improvements take the previous stage's results as benchmark. RFE performed with LR achieved Recall of 63.01 (0.04% increase), and Precision of 81.31 (3.94% decrease). DT performed with ANOVA-F FS slightly improved its Recall to 77.86 (1.89% increase) and its Precision to 78.05 (1.77% increase). RF's Recall further increased with almost all techniques; however, the most significant advancement was performed by the combination of SMOTE and RFE, resulting in Recall of 83.36 (6.7% increase) at the expense of a 7.35% decline in Precision. NB's Recall of 80.09 (5.63% increase) and Precision of 16.24 (6.35% increase) was achieved by a combination of ADASYN resampling and ANOVA-F FS. At last, KNN applied with ANOVA-F seen an 89.03% increase in Recall score (55.71), and a 40.49% increase in Precision (96.18). These results are a 523% and 14% increase in Precision and Recall, respectively, on the results obtained on raw data.

	Recall	Recall incr.	Precision	Precision incr.
LR - RFE	63.01	0.04%	81.31	-3.94%
DT - ANOVA-F	77.86	1.89%	78.05	1.77%
RF - SMOTE + RFE	83.36	6.70%	88.46	-7.35%
NB - ADASYN + ANOVA-F	80.09	5.63%	16.24	6.35%
KNN - ANOVA-F	55.71	89.03%	96.18	40.49%

Table 3. Best Performing Combinations of Resampling and FS

Stage 4

As Random Forest performed best across all stages of the proposed method, the ensemble model experimentation primarily focused on whether its results can possibly be further improved by combining it with different algorithms.

The paper first analysed RF's best configuration, that is, SMOTE oversampling and RFE feature selection. As in the same model setup no other algorithms outperformed RF either in Recall or Precision, the paper pursued alternative approaches.

First, it tried to find the algorithms that did better than RF in terms of Precision, expecting their combination to result in even further improved metrics. KNN, which achieved the highest Precision of 96.18 with TomekLinks and ANOVA-F was combined with RF. Their ensemble model gave the highest score of 97.95, but Recall dropped to 53.88. This result initially indicated that there is a trade-off between improved Precision at the expense of declining Recall. Another ensemble with RF, DT, KNN and ANOVA-F FS seemed to validate this thought. Whilst both RF and KNN had outstanding Precision scores, the fact that apart from RF, the other two had lower Recall scores resulted in high Precision (94.54), but an average Recall of 77.45. Further models were implemented (including a combination of all algorithms, a subset of best performing classifiers, etc.), but they all suggested that RF's Recall cannot be improved upon this way, and in particular, with a Simple Majority Voting setup of ensembles.

Therefore, the paper turned to 'soft' Weighted Majority Voting ensembles in an attempt to give more importance to RF in the model, but at the same time, keep other algorithms in place for a more robust performance. After a number of tries, RF combined with Logistic Regression and Random Oversampling, with ensemble weights 2:1 to RF, gave a Recall score of 83.56, Precision of 89.68, and F1-Score of 86.39 – all three metrics are the highest across the whole experiment.

	Accuracy	Precision	Recall	F1-Score	ROC-AUC
DT + RF / RUS / RFE	0.9995	0.9524	0.7298	0.8261	0.8649
RF + KNN / TomekLinks / ANOVAF	0.9992	0.9795	0.5388	0.6463	0.7694
RF + DT + LR / SMOTE / RFE	0.9991	0.7314	0.8295	0.776	0.9144
RF + DT + LR + NB + KNN/SMOTE/RFE	0.9991	0.7452	0.8254	0.7798	0.9124
RF + DT + LR + KNN/TomekLinks/RFE	0.9994	0.9443	0.7521	0.8366	0.876
RF + DT + KNN /ANOVAF	0.9995	0.9454	0.7745	0.8511	0.8872
RF + KNN/ANOVAF [3:1]	0.9995	0.9574	0.7684	0.8517	0.8841
RF + KNN/ANOVAF [2:1]	0.9995	0.9575	0.7522	0.8403	0.876
RF + LR/ROS [2:1]	0.9995	0.8968	0.8356	0.8639	0.9177

Table 4. Ensemble Model Results

Overall, the above RF-LR/ROS ensemble Recall result is an 8.46% performance improvement compared to the benchmark Recall score, that Random Forest achieved on raw data. It also compares well with the values seen from various literature, whether if the comparison is on only resampling, feature selection, ensembles, or on some combination of them.

Looking through all model configurations with respect to the five algorithms analysed, there are a few overarching patterns to note. First, as mentioned before, all classifiers' Recall scores drastically improved at the first stage after resampling the data. Interestingly, it is not specific to any under- or oversampling methods, all six resampling techniques performed well. At the same time, undersampling techniques RUS and NearMiss introduced a heavy precision-recall trade-off with all classifiers, essentially tagging all data samples as fraud. Contrarily, TomekLinks gave similar results for all classifiers as what they obtained from raw data. ROS, SMOTE, ADASYN also introduced a similar trade-off, but not as extreme as RUS and NearMiss. RF performed well with all three oversampling techniques.

Second, feature selection techniques on their own produced better results in most cases than what was achieved simply with resampling. Once paired with resampling methods, the precision-recall imbalance introduced earlier remained throughout most configurations, although Recall scores further improved.

Finally, based on the Random Forest ensemble model results, it seems that simply by combining conceptually different algorithms on resampled data can produce as much or even more improvements on benchmark scores than when resampling is complemented with feature selection techniques. Nonetheless, all stages of the experiment were needed to identify the approximate

configurations required in order to attain the best model, from algorithm selection through weights assignment, and resampling technique.

Future Work

Further research could be conducted on how hyperparameter optimisation of the discussed algorithms and cross-validation method would affect results. Additionally, new resampling and feature selection techniques could be added to the experiment.

REFERENCES

Al-Hashedi KG., Magalingam P. (2021). Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Comput. Sci. Rev. 2021, 40, 100402.*

Alfaiz N.S., Fati S.M. (2022). Enhanced Credit Card Fraud Detection Model Using Machine Learning. *Electronics 2022, 11, 662.*

Awoyemi Jo., Adetunmbi AO., Oluwadare SA. (2017). Credit card fraud detection using machine learning techniques: a comparative analysis. In: International Conference on Computer Networks and Information (ICCNI), pp. 1-9.

Biswas M., Debbarma S. (2022). *A new approach for credit card fraud detection using Machine Learning.* THEETAS 2022, April 16-17, Jabalpur, India.

Boehmke B., Greenwell B.M. (2019). *Hands-on Machine Learning with R (1st ed.).* Chapman and Hall/CRC.

Bouziane H., Messabih B., Chouarfia A. (2011). *Profiles and majority voting-based ensemble method for protein secondary structure prediction. Evol Bioinform Online. 2011, 7:171-89. Doi: 10.4137/EBO.S7931. Epub 2011 Oct.10. PMID: 22058650.*

Chandola V., Banerjee A., Kumar V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys, Vol. 41, No. 3, Article 15, pp. 1-58.*

Chandrashekar G., Sahin F. (2014). *A survey on feature selection methods.* Computers & Electrical Engineering, Vol. 40, Issue 1, pp. 16-28.

Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.* Journal of Artificial Intelligence Research, Vol. 16, pp. 321-357.

Chen X., Jeong J.C., (2007). *Enhanced recursive feature elimination.* Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pp. 429-435.

Devi R.G., Sumanjani P. (2015). *Improved classification techniques by combining KNN and Random Forest with Naïve Bayesian classifier.* 2015 International Conference on Engineering and Technology (ICETECH), pp. 1-4.

Dhankhad B., Far E., Mohammed A., Far B. (2018). Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study. *In: 2018 IEEE International Conference on Information Reuse and Integration (IRI), pp. 122-125. IEEE.*

Dornadula Vn., Geetha S. (2019). Credit card fraud detection using machine learning algorithms. Proc Computer Science; 165:631-641.

Dubey R., Zhou J., Wang Y., Thompson P.M., Ye J. (2015). *Analysis of Sampling Techniques for Imbalanced Data: AN N=648 ADNI Study.* Alzheimer's Disease Neuroimaging Initiative.

Esra M.K., Özel S.A., Ībrikçi T. (2012). *A comparative study on the effect of feature selection on classification accuracy.* Procedia Technology 1 (2012), pp. 323-327.

Faraji Z. (2022). A Review of Machine Learning Applications for Credit Card Fraud Detection with A Case study. *SEISENSE Journal Management, Vol. 5, No 1., pp. 49-59.*

Galar M., Fernandez A., Barrenechea E., Bustince H., Herrera F. (2012). *A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches.* In IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 42, No. 4, pp. 463-484.

Gonaboina H., Muttipati A.S. (2021). Machine learning methods for Discovering Credit Card Fraud. *Int. Res. J. Comput. Sci., 8, 1-6.*

Google Colab [Online]. Available at: https://colab.research.google.com/

He H., Bai B., Garcia E.A., Li S. (2008). *ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning.* IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322-1328.

Hofmann D.H. (1994). *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. Available at: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

IEEE Computational Intelligence Society. (2019). *IEEE-CIS Fraud Detection: Can you Detect Fraud from Customer Transactions?* Available at: https://www.kaggle.com/c/ieee-fraud-detection/overview.

Ileberi E., Sun Y., Wang Z. (2022). *A machine learning based credit card fraud detection using GA algorithm for feature selection.* Journal of Big Data, Volume 9:24.

John H., Naaz S. (2019). Credit card fraud detection using local outlier factor and isolation forest. *Int. J. Comput. Sc. Eng. 2019, 7, 1060-1064.*

Kaggle. (2021). *Anonymised credit card transactions labelled as fraudulent or genuine.* Machine Learning Group – ULB. Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

Kasongo S.M. (2021). *An Advanced Intrusion Detection System for IIoT Based on GA and Tree Based Algorithms.* IEEE Access PP(99):1-1.

Khare N., Sait S.Y. (2018). Credit card fraud detection using machine learning models and collating machine learning models. Int J Pure Applied Math, 118(20):825-38.

Khatri S., Arora A., Agrawal AP. (2020). Supervised machine learning algorithms for credit card fraud detection: a comparison. In: 10th International Conference on Cloud Computing, Data Science & Enginerring (Confluence), pp. 680-683.

Kumar M., Soundarya V., Kavitha S., Keeerthika E., Aswini E. (2019). Credit card fraud detection using random forest algorithm. *In Proceedings of the 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 21-22 February 2019, pp. 149-153.*

Liu FT., Ting KM., Zhou ZH. (2008). Isolation Forest. In Proceedings of the 2008 Eight IEEE International Conference on Data Mining, Ballarat, VIC, Australia, 15-19 December 2008, pp. 413-422.

Lucas Y., Jurgovsky J. (2020). Credit card fraud detection using machine learning: A survey.

Malik E.F., Khaw K.W., Wong W.P., Chew X. (2022). Credit Card Fraud Detection Using a New Hybrid Machine Learning Architecture. *Mathematics 2022, 10, 1480.*

Mani I., Zhang, J. (2003). *kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction.* Proceeding of International Conference on Machine Learning (ICML), Workshop on Learning from Imbalanced Data Sets, Washington DC.

Mishra A., Ghorpade C. (2018). Credit Card Fraud Detection on the Skewed Data Using Various Classification and Ensemble Techniques. *In: 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-5. IEEE.*

Nilson Report. (2021). *Card Fraud Losses Dip to $28.58 Billion.* Available at: https://nilsonreport.com/mention/1515/1link/.

Pozzolo A.D., Caelen O., Borgne Yann-Aël B., Waterschoot S., Bontempi G. (2014). *Learned lessons in credit card fraud detection from a practitioner perspective.* Expert Systems with Applications, Vol. 41, pp. 4915-4928.

Puh M., Brkic L. (2019). Detecting credit card fraud using selected machine learning algorithms. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Zagreb, Croatia, 20-24 May, pp. 1250-1255.

Randhawa K., Loo CK., Seera M., Lim CP., Nandi AK. (2018). Credit Card Fraud Detection Using AdaBoost and Majority Voting. *IEEE Access 2018, 6, 14277-14284.*

Saheed YK., Hambali MA., Arowolo MO., Olasupo YA. (2020). Application of GA feature selection on Naïve Bayes, Random Forest and SVM for credit card fraud detection. In: 2020 International Conference on Decision Aid Sciences and Application (DASA), pp. 1091-1097.

Scikit-learn: machine learning in Python [Online]. Available at: https://scikit-learn.org/stable/

Sivanantham S., Dhinagar SR., Kawin PA., Amarnath J. (2021). Hybrid Approach Using Machine Learning Techniques in Credit Card Fraud Detection. *In: Advances in Smart System Technologies, Springer: Singapore, 2021.*

Song Y-y., Lu Y. (2015). *Decision tree methods: applications for classification and prediction.* Shanghai Arch Psychiatry. 2015 Apr 25;27(2):130-5.

Stojanovic B., Bozic J., Hofer-Schmitz K., Nahrgang K., Weber A., Badii A., Sundaram M., Jordan E., Runevic J. (2021). *Follow the Trail: Machine Learning for Fraud Detection in Fintech Applications. Sensors 2021, 21, 1594.*

Swana E.F., Doorsamy W., Bokoro P. (2022). *Tomek Link and SMOTE Approaches for Machine Fault Classification with an Imbalanced Dataset.* Sensors, Vol. 22(9), 3246.

Taha A.A., Malebary S.J. (2020). An intelligent approach to credit card fraud detection using an optimised light gradient boosting machine. *IEEE Access 2020, 8, 25579-25587.*

Thennakoon A., Bhagyani C., Premadasa S., Mihiranga S. (2019). *Real-time Credit Card Fraud Detection Using Machine Learning.* Conference: International Conference on Cloud Computing, Data Science & Engineering (Confluence), at: India.

Thudumu S., Branch P., Jin J., Singh J.J. (2020). *A comprehensive survey of anomaly detection techniques for high dimensional big data.* Survey Paper, in: Journal of Big Data, 7:42.

Tsai C.F., Lin W.C. (2021). Feature selection and ensemble learning techniques in one-class classifiers: An empirical study of two-class imbalanced datasets. *IEEE Access 20201, 9, 13717- 13726.*

Varmedja D., Karanovic M., Sladojevic S., Arsenovic M., Anderla A. (2019). Credit card fraud detection-machine learning methods. In: 18th International Symposium INFOTEH-JAHORINA (INFOTEH), pp. 1-5.

Vengatesan K., Kumar A., Yuvraj S., Kumar V., Sabnis S. (2020). Credit card fraud detection using data analytic techniques. *Adv. Math. Sci. J. 2020, 9, 1185-1196.*

Yellowbrick. *Recursive Feature Elimination.* Available at: https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html.

Appendix

LR	Raw Data	RUS	NearMiss	TomekL	ROS	SMOTE	ADASYN
Accuracy	0.9991	0.9547	0.8015	0.9991	0.9642	0.98	0.9723
Precision	0.8451	0.0334	0.0084	0.8465	0.0423	0.0789	0.054
Recall	0.62	0.8924	0.933	0.6298	0.9066	0.8823	0.8803
F1-Score	0.7137	0.0644	0.0168	0.7198	0.0809	0.1438	0.1017
ROC-AUC	0.8099	0.9236	0.8671	0.8148	0.9355	0.9312	0.9264

	RUS + ANVAF	NearMiss + ANVAF	TomekL + ANVAF	ROS + ANVAF	SMOTE + ANVAF	ADASYN + ANVAF	ANVAF - RAW
Accuracy	0.9652	0.7024	0.9991	0.9661	0.9775	0.9761	0.9991
Precision	0.0434	0.0172	0.8699	0.0441	0.0658	0.0668	0.8688
Recall	0.8985	0.9331	0.6015	0.9005	0.8782	0.8802	0.6097
F1-Score	0.0827	0.0333	0.7097	0.0841	0.1223	0.1234	0.715
ROC-AUC	0.9319	0.8175	0.8007	0.9334	0.9279	0.9283	0.8047

	RUS + RFE	NearMiss + RFE	TomekL + RFE	ROS + RFE	SMOTE + RFE	ADASYN + RFE	RFE - RAW
Accuracy	0.9616	0.8062	0.9991	0.9595	0.9662	0.9783	0.999
Precision	0.0435	0.0086	0.8534	0.0426	0.0427	0.0727	0.8131
Recall	0.9148	0.9249	0.6179	0.9005	0.864	0.8904	0.6301
F1-Score	0.0829	0.017	0.714	0.0811	0.0815	0.1339	0.6992
ROC-AUC	0.9382	0.8655	0.8089	0.93	0.9152	0.9344	0.8149

Table 1. Logistic Regression Classifier Model Results

DT	Raw Data	RUS	NearMiss	TomekL	ROS	SMOTE	ADASYN
Accuracy	0.9991	0.8896	0.5282	0.9991	0.999	0.9976	0.9977
Precision	0.7534	0.0143	0.0038	0.7669	0.7478	0.4111	0.4196
Recall	0.7603	0.9147	0.9452	0.7641	0.7237	0.7787	0.7725
F1-Score	0.7555	0.0282	0.0076	0.7643	0.7345	0.5372	0.543
ROC-AUC	0.8799	0.9021	0.7363	0.8818	0.8616	0.8884	0.8853

	RUS + ANVAF	NearMiss + ANVAF	TomekL + ANVAF	ROS + ANVAF	SMOTE + ANVAF	ADASYN + ANVAF	ANVAF - RAW
Accuracy	0.9051	0.3479	0.9992	0.9991	0.9975	0.9977	0.9992
Precision	0.0163	0.003	0.7858	0.7762	0.3987	0.4286	0.7805
Recall	0.9087	0.9655	0.7704	0.7358	0.7828	0.8071	0.7786
F1-Score	0.0322	0.006	0.7768	0.7538	0.5278	0.5573	0.7783
ROC-AUC	0.9069	0.6562	0.885	0.8677	0.8903	0.9026	0.8891

	RUS + RFE	NearMiss + RFE	TomekL + RFE	ROS + RFE	SMOTE + RFE	ADASYN + RFE	RFE - RAW
Accuracy	0.9036	0.5256	0.9991	0.9991	0.9977	0.9976	0.9991
Precision	0.0162	0.0037	0.8633	0.7562	0.4208	0.4066	0.759
Recall	0.9107	0.9471	0.6342	0.7379	0.7868	0.7685	0.7726
F1-Score	0.0318	0.0075	0.7284	0.7466	0.5477	0.5301	0.7644
ROC-AUC	0.9071	0.736	0.817	0.8687	0.8925	0.8832	0.886

Table 2. Decision Tree Classifier Model Results

RF	Raw Data	RUS	NearMiss	TomekL	ROS	SMOTE	ADASYN
Accuracy	0.9995	0.9769	0.7993	0.9995	0.9995	0.9995	0.9994
Precision	0.9526	0.0667	0.009	0.9548	0.9453	0.8796	0.8817
Recall	0.7704	0.9086	0.929	0.7812	0.7725	0.8274	0.8213
F1-Score	0.8514	0.124	0.0178	0.858	0.8496	0.8517	0.8491
ROC-AUC	0.8851	0.9428	0.864	0.8906	0.8862	0.9136	0.9105

	RUS + ANVAF	NearMiss + ANVAF	TomekL + ANVAF	ROS + ANVAF	SMOTE + ANVAF	ADASYN + ANVAF	ANVAF - RAW
Accuracy	0.9747	0.3882	0.9995	0.9995	0.9994	0.9995	0.9995
Precision	0.0584	0.0058	0.9461	0.9436	0.8726	0.8778	0.9528
Recall	0.8964	0.9716	0.7846	0.7847	0.8315	0.8315	0.7806
F1-Score	0.1097	0.0116	0.8576	0.8564	0.8506	0.8532	0.8578
ROC-AUC	0.9356	0.6794	0.8922	0.8923	0.9156	0.9156	0.8902

	RUS + RFE	NearMiss + RFE	TomekL + RFE	ROS + RFE	SMOTE + RFE	ADASYN + RFE	RFE - RAW
Accuracy	0.969	0.7282	0.9995	0.9995	0.9995	0.9995	0.9995
Precision	0.0499	0.0072	0.9488	0.9449	0.8846	0.8867	0.9526
Recall	0.9107	0.9492	0.7826	0.7725	0.8336	0.8254	0.7765
F1-Score	0.0944	0.0143	0.8571	0.8496	0.8568	0.854	0.8553
ROC-AUC	0.9399	0.8385	0.8912	0.8862	0.9167	0.9126	0.8882

Table 3. Random Forest Classifier Model Results

NB	Raw Data	RUS	NearMiss	TomekL	ROS	SMOTE	ADASYN
Accuracy	0.9928	0.986	0.7914	0.9928	0.9904	0.9923	0.9919
Precision	0.1467	0.0893	0.0067	0.1454	0.1231	0.1527	0.149
Recall	0.6462	0.7316	0.7865	0.6471	0.7378	0.7582	0.7745
F1-Score	0.239	0.1587	0.0133	0.2371	0.211	0.2541	0.2499
ROC-AUC	0.8198	0.859	0.789	0.8202	0.8643	0.8754	0.8834

	RUS + ANVAF	NearMiss + ANVAF	TomekL + ANVAF	ROS + ANVAF	SMOTE + ANVAF	ADASYN + ANVAF	ANVAF - RAW
Accuracy	0.9806	0.8651	0.9854	0.9919	0.9921	0.9925	0.9854
Precision	0.0868	0.0502	0.11	0.1492	0.1528	0.1624	0.11
Recall	0.8213	0.8559	0.7644	0.7805	0.7826	0.8009	0.7644
F1-Score	0.1534	0.0886	0.1847	0.2506	0.2556	0.27	0.1847
ROC-AUC	0.9011	0.8605	0.8751	0.8864	0.8875	0.8969	0.8751

	RUS + RFE	NearMiss + RFE	TomekL + RFE	ROS + RFE	SMOTE + RFE	ADASYN + RFE	RFE - RAW
Accuracy	0.9794	0.7743	0.9896	0.9847	0.9933	0.9936	0.9894
Precision	0.0815	0.0063	0.1578	0.1012	0.1843	0.191	0.1543
Recall	0.8252	0.8007	0.754	0.7845	0.7867	0.7989	0.717
F1-Score	0.1448	0.0125	0.2503	0.1759	0.297	0.3075	0.242
ROC-AUC	0.9024	0.7875	0.872	0.8848	0.8902	0.8964	0.8535

Table 4. Naïve Bayes Model Results

KNN	Raw Data	RUS	NearMiss	TomekL	ROS	SMOTE	ADASYN
Accuracy	0.9983	0.6601	0.0493	0.9983	0.9976	0.9571	0.9554
Precision	0.8413	0.0033	0.0017	0.8576	0.2628	0.0216	0.0207
Recall	0.0894	0.6525	0.9409	0.0871	0.2134	0.5387	0.5387
F1-Score	0.16	0.0066	0.0034	0.1575	0.2352	0.0415	0.04
ROC-AUC	0.5446	0.6563	0.4943	0.5435	0.6061	0.7483	0.7474

	RUS + ANVAF	NearMiss + ANVAF	TomekL + ANVAF	ROS + ANVAF	SMOTE + ANVAF	ADASYN + ANVAF	ANVAF - RAW
Accuracy	0.8277	0.319	0.9991	0.9985	0.984	0.9828	0.9991
Precision	0.0311	0.0031	0.9618	0.6846	0.0768	0.0707	0.9618
Recall	0.799	0.9411	0.5571	0.2947	0.7442	0.734	0.5571
F1-Score	0.0589	0.0063	0.6502	0.4115	0.1392	0.1289	0.6502
ROC-AUC	0.8133	0.6295	0.7785	0.6472	0.8643	0.8586	0.7785

	RUS + RFE	NearMiss + RFE	TomekL + RFE	ROS + RFE	SMOTE + RFE	ADASYN + RFE	RFE - RAW
Accuracy	0.7694	0.0491	0.9988	0.998	0.9622	0.9611	0.9894
Precision	0.0192	0.0017	0.8915	0.4351	0.032	0.0314	0.1543
Recall	0.7606	0.9389	0.3534	0.4186	0.565	0.5608	0.717
F1-Score	0.0368	0.0034	0.4419	0.425	0.0597	0.0586	0.242
ROC-AUC	0.765	0.4932	0.6767	0.7088	0.7639	0.7613	0.8535

Table 5. K-Nearest Neighbour Classifier Model Results

MSc Project - Reflective Essay

Project Title:	Credit Card Fraud Detection Using Supervised Machine Learning Algorithms – Experiments with Resampling, Feature Selection, and Ensemble Models
Student Name:	Adam Toth
Student Number:	210505924
Supervisor Name:	Ms Zunaira Nadeem
Programme of Study:	MSc FT Computer Science (Conversion) (PMSF-QMCOMP2)

Overview

My research paper attempted to achieve two main objectives. First, I noticed that in the existing literature that I encountered around the highly class imbalanced European Cardholder Credit Card Fraud Detection Dataset, have not proposed a model architecture in which resampling, feature selection techniques, and ensemble models were all present. Most authors work with either one or two of these, and even then, experimental results for multiple techniques within resampling or FS are not too common. It made me curious whether the most common resampling (e.g., SMOTE) or FS techniques (e.g., Genetic Algorithm, RFE) actually give the best results, or could be improved upon by simply stacking these techniques on each other (in addition to combining several models together in ensembles).

Therefore, this project first analyses five supervised machine learning algorithms (Logistic Regression, Decision Tree, Random Forest, Naïve Bayes, and K-Nearest Neighbours) without any resampling or FS, providing benchmark evaluation metrics for later stages. The paper then considers several resampling techniques (Undersampling: Random Under Sampling, Near Miss and Tomek Links Undersampling; Oversampling: Random Over Sampling, Synthetic Minority Oversampling Technique and Adaptive Synthetic Oversampling) next at the second stage. At the third stage, a filter- and wrapper-based FS technique (ANOVA-F, Recursive Feature Elimination) is analysed first on raw data, then in combination with all the resampling techniques considered at the second stage. At the final, fourth stage, the best performing algorithm from the third stage is combined with the second and third best algorithms in ensembles.

The abovementioned systematic approach achieves the second objective of the paper, which is the observation of possible incremental improvements in results of the classifiers at each consecutive steps.

Analysis of strengths/weaknesses

Strengths

- Whilst the benchmark results of the classifiers on raw data are predictably very similar to those found in the literature, further stages of the experiment produced quite outstanding evaluation scores from unexpected combination of the above techniques.
- The experiment is also systematic in its approach. More than 110 models are trained and tested which allows insights to be gained from multiple angles.

Weaknesses

- Hyperparameter tuning (although was not in the scope of the objectives of the paper) was not performed at all, which could have seriously improved scores further. It would have been also nice to see more resampling and feature selection techniques.
- The code blocks that I used are not necessarily the most computationally efficient ones. Even without hyperparameter optimisation, training times seriously limited the amount of work that I managed to accomplish. If I had more time, I could have experimented even more with the ensemble models – that part of the research lacked a rigorous approach and depended more on intuition gained from the previous stages.

Presentation of possibilities for further work
As mentioned in the paper and in 'Weaknesses', the existing experiment could benefit from hyperparameter tuning. Building on those results and set-up, unsupervised algorithms could be performed on the same dataset and compared with the current algorithms.

Critical analysis of the relationship between theory and practical work produced
I believe that the work produced, and results obtained in this experiment are very similar those found in the literature, at least in terms of structure and rigour. The theoretical guidelines were implemented throughout the paper, whether it was creating benchmark results performed on raw data and metrics drawn from the literature (using the same dataset), using an advanced cross-validation technique (stratified k-fold), and placing each step of the experiment in a pipeline that prevents data leakage from the training folds towards the test folds. Appropriate metrics were selected for model comparison that recognise the imbalanced nature of the data and (partially) reflect the business context of the problem.

However, probably the most interesting part of this project was the realisation that the literature I read primarily focused on maximising some selected metrics without actual justification for it (which I followed). However, as mentioned in the paper, given specific financial costs associated with, for example low recall or precision values, one could define the financially most beneficial threshold values for each metrics, one that maximises utility of a financial institution. For example, contrary to my argument made in the paper siding with recall as a metric that should be prioritised, a report made by Clearsale (2020) stated that losses due to false positives were projected to reach $443 billion in 2021 in the e-commerce sector, whilst losses due to actual fraud were projected to be about $6.4 billion (more than 70 times less). Considering these figures, KNN should have been selected with ANOVA-F feature selection as its precision outperforms all other combinations in the experiment, despite its lagging recall score.

Awareness of Legal, Social, Ethical Issues and Sustainability
The dataset I used for this paper was anonymised due to privacy concerns, hence any infringement of law can be ruled out.

References
Clearsale. (2020). *The Ecommerce Conundrum: Balancing False Declines and Fraud Prevention.* Available at: https://offer.clear.sale/false-declines-ecommerce-fraud-prevention-report-download-typ?submissionGuid=7c54d9d1-7a8d-449c-adec-e47bfff84a2c.