# *Comparative Study on Classical Machine Learning Algorithms for Credit Card Fraud Detection*

Mohammad Asim Iqbal
*Faculty of Engineering*
*Faculty of Science*
*University of Western Ontario*
London ON, Canada
miqbal57@uwo.ca

Waqar Bin Kalim
*Faculty of Science*
*University of Western Ontario*
London ON, Canada
wkalim@uwo.ca

Mohammad Ali Sarfraz
*Faculty of Engineering*
*Faculty of Science*
*University of Western Ontario*
London ON, Canada
msarfra2@uwo.ca

***Abstract*— Credit-cards have become one of the most popular means of performing monetary transactions in the modern-day markets, including online webstores. A major issue that comes with these forms of payments is that of fraudulent purchases because of a lost or stolen card, which may see the owner lose hundreds and thousands of dollars without their knowledge. Due to the nature of the datasets being biased towards non-fraudulent cases, a majority of classical machine learning algorithms cannot accurately model a credit-card fraud detection system in terms of authenticity. Thus, this study aims to define and utilize alternate performance metrics, like precision and recall, for evaluating the performance of various classical machine learning algorithms such as logistic regression, polynomial regression, support vector machines, and random forests, tuned to their optimal conditions. These models are then tested using the alternative evaluation metrics for classifying fraudulent cases in the biased dataset. The evaluation shows that the best algorithm to use for identifying fraudulent credit-card uses is the random forest, which yielded a maximum precision rating of 78.5%.**

***Keywords—credit-card, fraud, machine learning, precision, recall, logistic regression, polynomial regression, support vector machine, random forest.***

## I. INTRODUCTION

One of the most common forms of monetary payments in this day and age is through the use of multi-purpose line of credits issued by a bank or other financial services. This line of credit allows clients to essentially borrow funds to spend on various amenities and services with the promise of paying back the borrowed sum of money along with any applicable fees and interest if necessary [1]. A credit card is one such line of credit, where a plastic or metal card may be used by a client to complete a payment through a line of credit, hereby eliminating the need to carry and use large sums of cash on hand.

While this form of payment has ultimately made the processing of monetary transactions much easier and more efficient, it isn't necessarily free of flaws either. One of the major drawbacks of credit-based transactions is that of credit card fraud, which encompasses all transactions being made through the credit card without the permission of the cardholder/owner, hereby making them unaware of the situation. These fraudulent transactions occur when someone steals the credit card itself, or the prevalent information necessary to operate it, such as the personal identification number (PIN), and uses the credit card through in-store or online purchases [2].

Statistics show that there are approximately 75 million credit cards in circulation through Canada alone, with 89% of the population owning at least one credit card [3]. These numbers confirm the claim that credit cards do constitute a bulk of the monetary transactions happening within the country, and hence are a key asset that gets abused for fraudulent transactions.

The objective of this project is to obtain a dataset relating to credit card transactions being fraudulent or not and using this to train various machine learning algorithms learned in the course material for classifying whether the transaction is fraudulent. The algorithms are implemented using appropriate compiling and computation tools like *Jupyter* being powered by the *Python* programming language, and then tuned accordingly to acquire the best models for each instance. Each algorithm's performance is compared to itself (by adjusting the hyperparameters, where applicable) and other algorithms to showcase how each one processes and showcases the results, along with the model's prediction performance.

Each algorithm's analysis produces computation metrics, which will be used to evaluate its performance against the rest. Hence an underlying goal of the project is also to compute which algorithm is the best one for realizing the credit card fraud detection, hereby highlighting the trade-off between ease of implementation/maintainability and reliability.

## II. RELATED WORK

The Canadian Bankers Association has stated that nearly 800 million dollars have been lost through credit card fraud, bringing the total number of accounts reporting at least one case of credit card fraud up by 71% [3].

Hence, the issue of credit card frauds has been prevalent for some time now, and will only get worse as more cards are brought into circulation, noting the fact that the statistics shown above only reflect Canadian transactions, and are much higher when considered on a global scale. As such, there have been various means and methods for detecting fraudulent credit transactions when they occur. These methods include, but are not limited to algorithms using clustering techniques, neural networks, naïve bayes classifiers, support vector machines, and decision trees [4]. Each of these algorithms has their own set of pros and cons and can be established further to improve prediction accuracies and computation times.

Other studies conducted by *J. Brownlee* have shown both the evaluation and classification of imbalanced binary data for systematically evaluating a suite of machine learning models using a robust test harness [5]. This project used the same dataset, but only assessed the fitting of the data for supervised learning algorithms like K-nearest-neighbour and decision trees as opposed to a general study of how each algorithm behaved upon tuning the hyperparameters.

Another project by *L. Frei* aims to relate the classic machine learning algorithm's performance to each other while identifying the biased nature of the dataset [6]. However, this study only shows comparisons in between the different algorithms and does not detail how tuning the penalty types or regularization methods affected the performance metrics.

## III. METHOD

This section highlights the procedure for the implementation of various machine learning algorithms and the computational metrics used for determining their overall effectiveness. Since the algorithm is going to be predicting whether a transaction is fraudulent or not, a suitable form of training and testing datasets are essential. For the purposes of this project, the *creditcard.csv* dataset was acquired from *Kaggle*, a free public online repository of community published data [7]. The dataset contains credit card transactions made over a period of two days by European cardholders in September of 2013, showcasing the 492 cases of reported fraudulent transactions in a total of 284,807 transactions.

The first stage of data processing must always be verification and validation for useability. As such, the data obtained from the file was scanned to check if there were any field missing or containing null values instead of appropriate numeric values (see *Figure 1*). With the data completeness verified, the next stage is to try and reduce the dimensionality of the data as much as possible for optimizing the computation times. However, performing a principal component analysis (PCA) on the data revealed that it had already undergone this feature, and therefore all 29 components are essential for capturing the variance in the data and reducing information loss. A possible explanation for the data already having gone through PCA is privacy, where an account/card number or owner name may have been filtered out due to their lack of impact on the overall outcome of the transaction. It should also be noted that this data is highly biased towards non-fraudulent transactions, which needs to be accounted for in the analysis.

```
In [6]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 30 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   V1      284807 non-null  float64
 1   V2      284807 non-null  float64
 2   V3      284807 non-null  float64
 3   V4      284807 non-null  float64
 4   V5      284807 non-null  float64
 5   V6      284807 non-null  float64
 6   V7      284807 non-null  float64
 7   V8      284807 non-null  float64
 8   V9      284807 non-null  float64
 9   V10     284807 non-null  float64
 10  V11     284807 non-null  float64
 11  V12     284807 non-null  float64
 12  V13     284807 non-null  float64
 13  V14     284807 non-null  float64
 14  V15     284807 non-null  float64
 15  V16     284807 non-null  float64
 16  V17     284807 non-null  float64
 17  V18     284807 non-null  float64
 18  V19     284807 non-null  float64
 19  V20     284807 non-null  float64
 20  V21     284807 non-null  float64
 21  V22     284807 non-null  float64
 22  V23     284807 non-null  float64
 23  V24     284807 non-null  float64
 24  V25     284807 non-null  float64
 25  V26     284807 non-null  float64
 26  V27     284807 non-null  float64
 27  V28     284807 non-null  float64
 28  Amount  284807 non-null  float64
 29  Class   284807 non-null  int64
dtypes: float64(29), int64(1)
memory usage: 65.2 MB
```

*Figure 1: Verification of data entries being non-null values.*

A total of seven supervised learning model algorithms have been researched, implemented and tuned to optimal functionality for the purposes of comparing their performance. A majority of these algorithms have been selected due to their inclusion in this course material, however some have been researched and implemented from outside the scope of the course as well. The algorithms chosen are listed below:

- Logistic Regression with No Regularization
- Logistic Regression with L1-Regularization
- Logistic Regression with L2 -Regularization
- 2nd Degree Polynomial Logistic Regression with No Regularization
- Gaussian SVM
- Polynomial SVM
- Random forest

Since the main objective is to compare the different types of algorithms in their ability to classify the fraudulent transactions, the actual implementation of the algorithm is not done explicitly in this project. As such, various Python libraries like pandas, *numpy*, and *sklearn* were imported to the workspace for getting access to predefined functions that apply the mentioned algorithms to the dataset for computing the performance metrics. As stated earlier, the dataset being used is highly biased towards predicting non fraudulent transactions on account of only 0.172% of the data being fraud cases. Therefore, the model's accuracy may not be used as a valid metric to evaluate its

performance, since the overwhelming data bias will cause it to predict a false by default and will defeat the purpose of the algorithm itself. Precision, recall/sensitivity, specificity, and F1-score are alternative metrics that will be used for model evaluation instead since these are not so easily influenced by the data bias. A summary highlighting each of these metrics in the context of machine learning is shown in *Table 1* below, note that the abbreviations are as follows:

- TP = True Positive.
- FP = False Positive.
- TN = True Negative.
- FN = False Negative.

*Table 1: Definitions of the computation metrics being evaluated for comparison purposes [8].*

| ***Computation Metric*** | ***Definition*** | ***Mathematical Formula*** |
|---|---|---|
| Accuracy | The number of correctly predicted outcomes from the total number of outcomes. | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| Precision | The number of positive predictions that were true positive outcomes. | $\frac{TP}{TP+FP}$ |
| Recall | The number of true positive outcomes predicted correctly. | $\frac{TP}{TP+FN}$ |
| F1-Score | Conveys the balance between the precision and recall metric values. | $2\times\frac{(Precision\times Recall)}{(Precision+Recall)}$ |

Using these performance definitions, the accuracy of the model cannot be solely used to compute its effectiveness because the performance is affected by the number of true negative predictions by the model, which provides minimal use to the bank or customer. On the other hand, the cost of a false negative is extremely high given its implications, and therefore needs to be represented by a metric that accounts for this event. Therefore, the recall value will be computed for each algorithm and has priority when ranking the algorithms. To a lesser degree, the consequences of a false positives can interfere with customer experience. The precision metric is used to account for such instances and give a better depiction of each algorithm's effectiveness, or proportion of correctness, in positive predictions. A balance between precision and recall provides a model that is sufficient in detecting fraud cases, while not making the credit card useless by flagging by constantly incorrectly identifying fraud cases. Thus, a combination of both precision and recall, which is expressed by the F1-score metric of the algorithms being implemented.

Given the large bias in the data, 5-fold cross validation is used (for hyperparameter tuning and overall algorithm performance) to have the test data approximate the same distribution as the training data. Algorithms with applicable hyperparameters are optimized based on the previously mentioned performance metrics. Once all the algorithms have been individually tuned to perform at their best, their evaluation metrics are compared with each other for identifying which algorithm is the most suitable for the purposes of credit-card fraud detection.

## IV. Experimental Results

This section highlights the plots showcasing the effects of varying the hyperparameters of appropriate algorithms, and also showcases snippets detailing the general algorithm performance and evaluation metrics as being computed by the machine. This allows users to compare the tuned algorithms with each other.

### *A. Algorithm Hyperparameter Tuning*

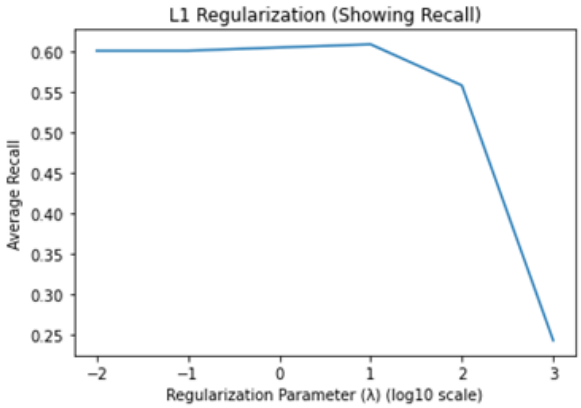


*Figure 2: Plot relating the average L1 regularization recall for various values of λ.*

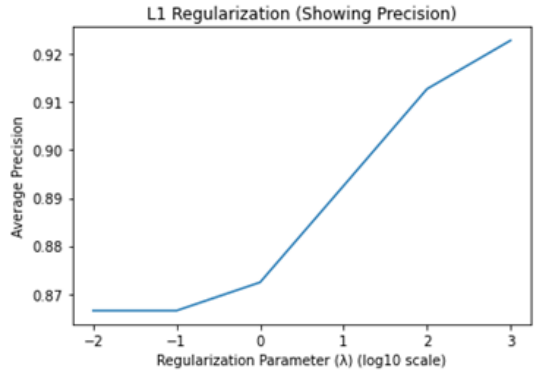


*Figure 3: Plot relating the average L1 regularization precision for various values of λ.*

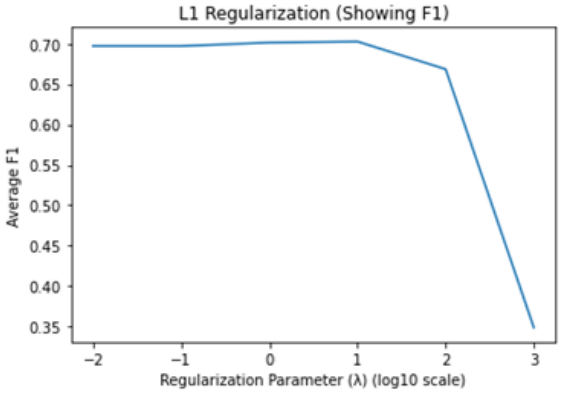


*Figure 4: Plot relating the average L1 regularization F1-score for various values of λ.*

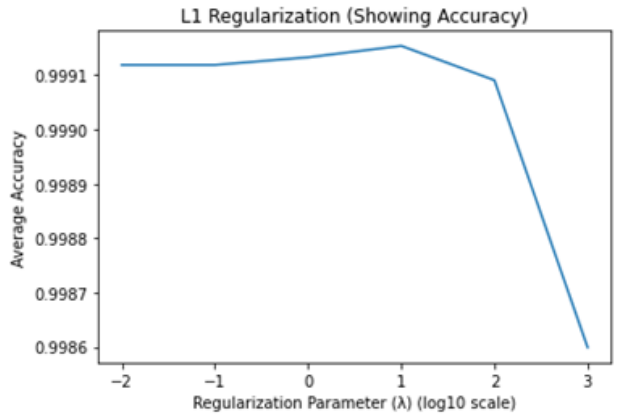


*Figure 5: Plot relating the average L1 regularization accuracy for various values of λ.*

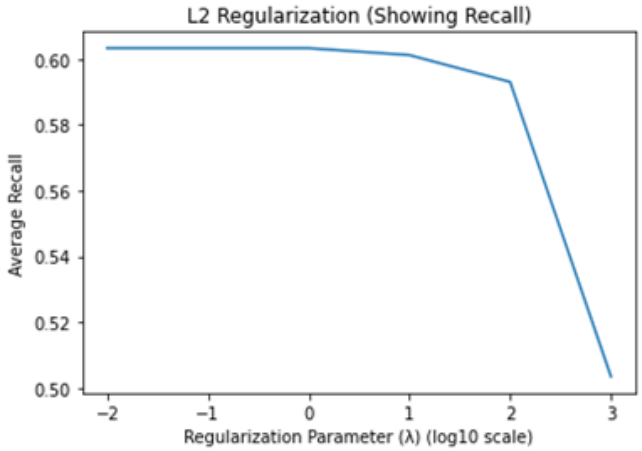


*Figure 6: Plot relating the average L2 regularization recall for various values of λ.*

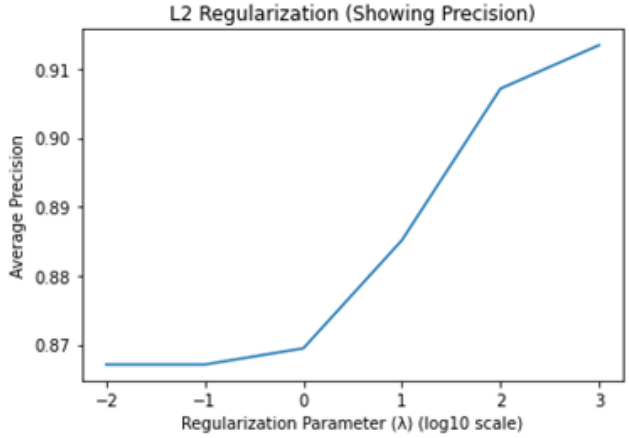


*Figure 7: Plot relating the average L2 regularization precision for various values of λ.*

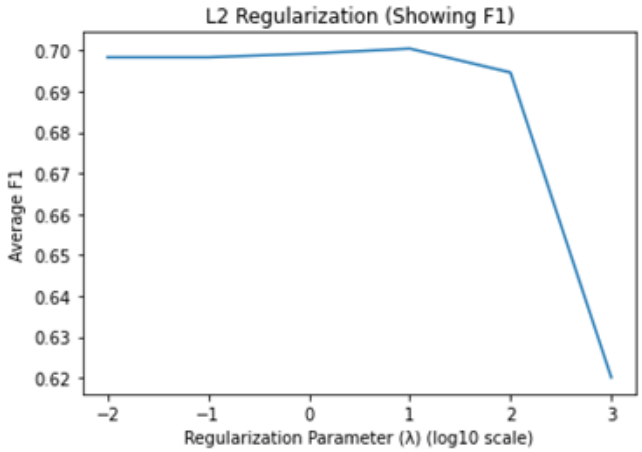


*Figure 8: Plot relating the average L2 regularization F1-score for various values of λ.*

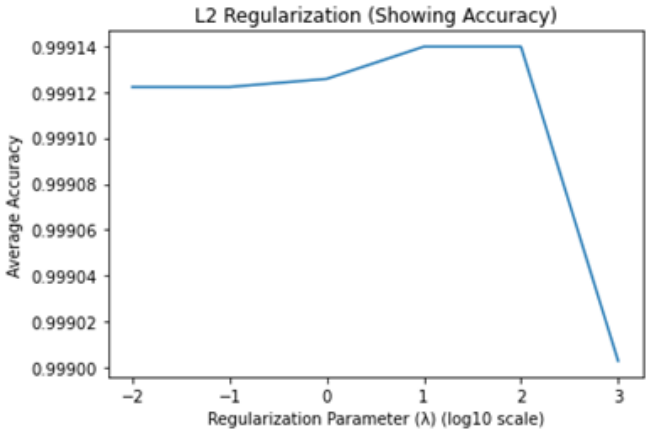


*Figure 9: Plot relating the average L2 regularization accuracy for various values of λ.*

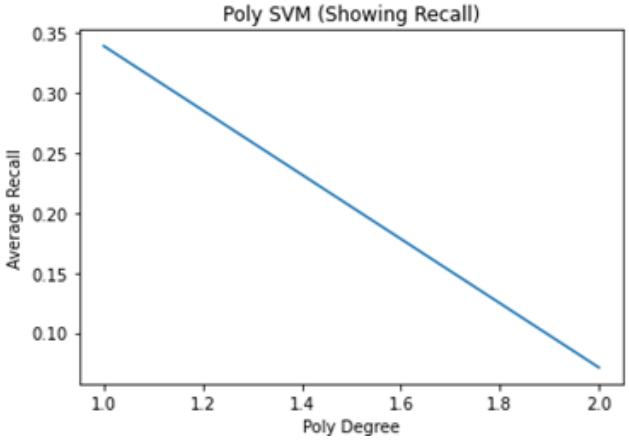


*Figure 10: Plot relating the average polynomial SVM recall for various degrees of the polynomial.*

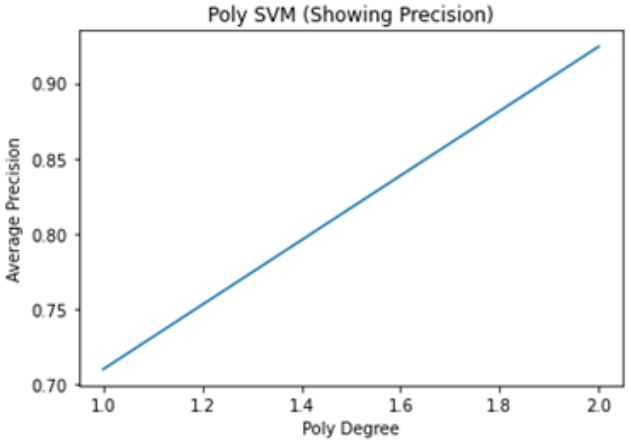


*Figure 11: Plot relating the average polynomial SVM precision for various degrees of the polynomial.*

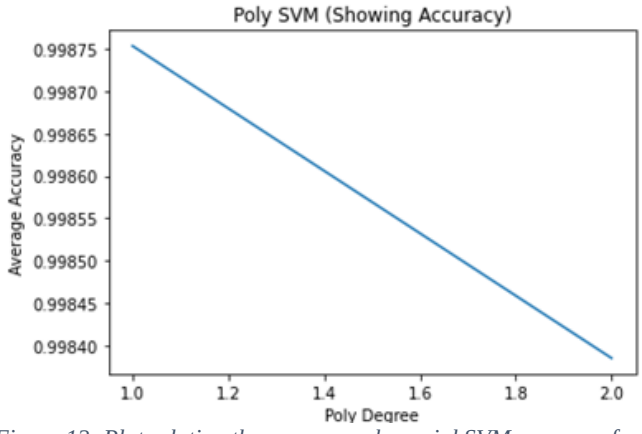


*Figure 12: Plot relating the average polynomial SVM accuracy for various degrees of the polynomial.*

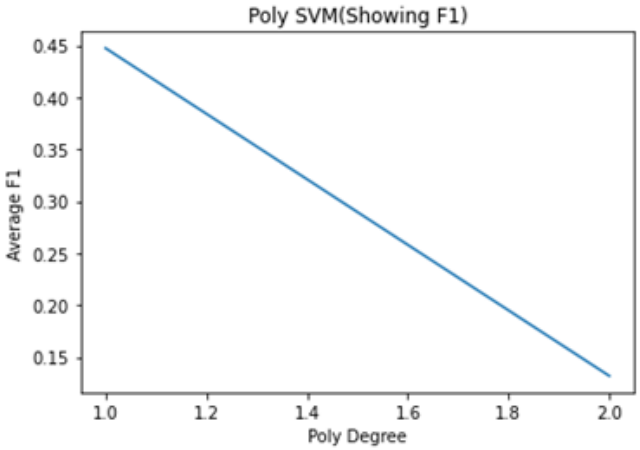


*Figure 13: Plot relating the average polynomial SVM F1-score for various degrees of the polynomial.*

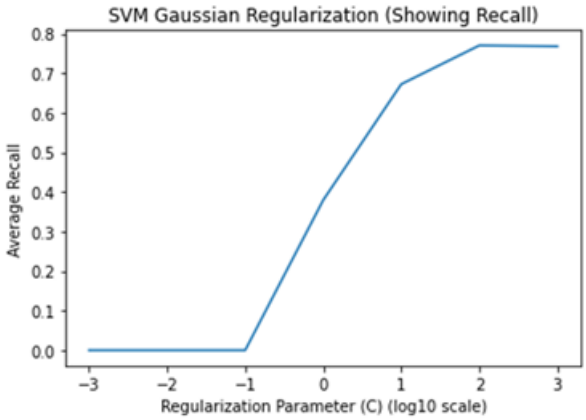


*Figure 14: Plot relating the average Gaussian SVM recall for various values of C.*

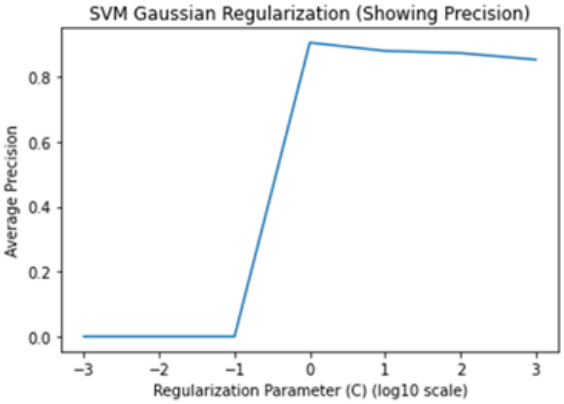


*Figure 15: Plot relating the average Gaussian SVM precision for various values of C.*

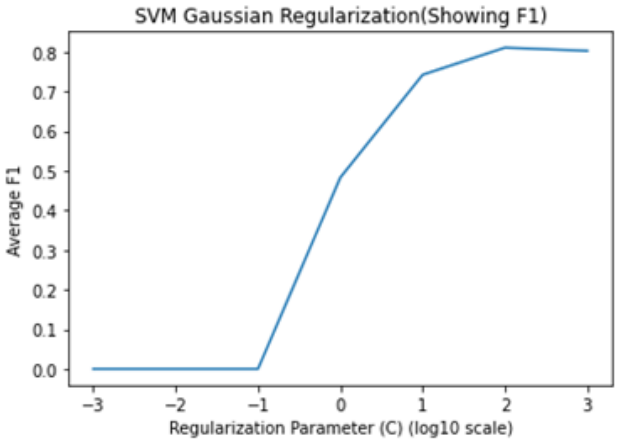


*Figure 16: Plot relating the average Gaussian SVM F1-score for various values of C*

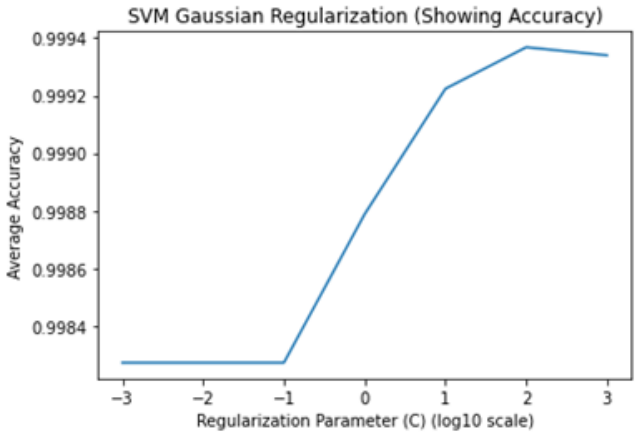


*Figure 17: Plot relating the average Gaussian SVM accuracy for various values of C.*

### *B. General Algorithm Performance Metrics*

Logistic Regression (No Regularization)

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 0.999 |
| Recall | 0.62 |
| Precision | 0.867 |
| F1-Score | 0.722 |

Logistic Regression (No Regularization) with 2nd Degree Polynomial Features

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 0.999 |
| Recall | 0.601 |
| Precision | 0.837 |
| F1-Score | 0.698 |

Logistic Regression with L1-Regularization and $\lambda = 10$

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 0.999 |
| Recall | 0.619 |
| Precision | 0.870 |
| F1-Score | 0.722 |

Logistic Regression with L2-Regularization and $\lambda = 1$

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 0.999 |
| Recall | 0.623 |
| Precision | 0.876 |
| F1-Score | 0.727 |

Polynomial SVM with Degree = 1

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 0.999 |
| Recall | 0.331 |
| Precision | 0.824 |
| F1-Score | 0.472 |

Gaussian SVM with Regularization Parameter $C = 100$

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 0.999 |
| Recall | 0.776 |
| Precision | 0.848 |
| F1-Score | 0.810 |

Random Forest with Default Parameters

| ***Metric*** | ***Value*** |
|---|---|
| Accuracy | 1.00 |
| Recall | 0.785 |
| Precision | 0.937 |
| F1-Score | 0.852 |

## V. EXPERIMENTAL ANALYSIS

### *A. Individual Algorithm Tuning Analysis*

#### *Logistic Regression (L1-Regularization) Varying Hyperparameter λ*

Observing the results from *Figures 2* to *5* above, varying the regularization parameter $\lambda$ had noticeable effects on certain performance metrics while minimal on others (*precision* and *accuracy*). *Recall* and *F1-score* (due to its relation to recall) were the most sensitive to change in $\lambda$, specifically when $\lambda$ was increased. This behaviour agrees with previously mentioned aspects of the data. In this case, due to the large amounts of non-fraudulent data, the model is more sensitive to positive cases, which consequently makes it less sensitive to the negative cases.

As the data being analysed primarily consists of non-fraudulent cases, a model that misclassifies non-fraudulent cases produces a larger total error than a model that misclassifies fraud cases. Meaning when the model becomes less flexible (reducing overfitting by increasing $\lambda$) it tends to conform to the overwhelming non-fraudulent cases. By increasing $\lambda$ the model flags fewer cases as positives and the cases it does flag as positive have a strong probability of being positive (given the model's bias towards classifying as negative), this ratio explains the slight increase in *precision*.

However, increasing $\lambda$ also tends to make the model misclassify actual positive cases, and given the context of the application, the cost of this false negative is extremely high. From *Figure 2* above it can be seen that recall does not vary for

the smaller values of $\lambda$, and only the larger values of $\lambda$ significantly impact the results as *recall* and *precision* begin to diverge (confirmed by *F1-score* in *Figure 4*). Therefore, a $\lambda$ value of *10* provides the best overall performance for this model since the recall is not reduced (main priority) and precision is marginally increased.

### *Logistic Regression (L2-Regularization) Varying Hyperparameter λ*

From *Figures 6* to *9*, it can be observed that *precision* and *accuracy* were generally insensitive to the change in $\lambda$, while *recall* and *F1-score* had a more significant impact as the value of $\lambda$ increased. Following similar explanations for these results as with the case of the L1-regularized logistic regression, when the penalty for misclassification becomes large, the bias in the data transfers over as a bias in the model. Larger values of $\lambda$ in the model produce marginally better *precision* scores, but drastically worse *recall* scores. Thus, larger values of $\lambda$ resulted in an overfitting of the biased training data, which in turn resulted in poor performance when identifying the occasional fraudulent case. Giving priority to higher *recall* scores, the best value for the hyperparameter $\lambda$ given the context of the overall application is 1. This value maintains the high *recall* score while marginally increasing the *precision* score.

### *Polynomial SVM Varying Degree of Polynomial*

Observing *Figures 10* to *13*, the SVM with a polynomial kernel is quite sensitive to change in degree. *Precision*, *recall*, and *F1-score* all had significant variations based on alteration of the polynomial degree, while *accuracy* was only marginally affected by the changes. The relationship between the results and increase of degree were not originally anticipated, as it was expected that increasing the polynomial degree would result in more flexible decision boundaries and thus better classification. However, based on both *Figures 10* and *11*, the *recall value* was negatively affected while *precision* was the opposite of this and saw a positive increase. Maybe add something explaining the results? The sample size of varying the polynomial degree was very small because of the computational/time requirements of higher degrees. This reason combined with a lack of performance computations resulted in the quadratic polynomial being the largest degree pursued for this project. Based on these results alone, a polynomial SVM of degree *1* (i.e. linear SVM) was chosen as the ideal candidate for this algorithm.

### *Gaussian SVM Varying Regularization Parameter C*

Based on Figures 14 to 17, the Gaussian SVM was generally very sensitive to variations in hyperparameter C. While very small values of C resulted in poor performance across the board (excluding accuracy) and insensitivity of the algorithm, C values larger than 0.1 resulted in drastic variations in the algorithm. Unlike previous algorithms, where varying the hyperparameter resulted in an inverse relationship between recall and precision, this one saw both recall and precision improve dramatically for larger values of C.

A possible explanation for this is that as C is increased, the penalty for misclassification is also increased. Hence, a finer decision boundary around the two classes occurs for large values of C, which allows the model to better capture the few positive cases (increasing the recall score). Additionally, the finer decision boundary may limit the model from predicting positive cases and as a result also increases the precision score. However, blatantly increasing C is not a viable option either as overfitting of the training data occurs and test performance diminishes, as evident in the precision, recall, and F1-scores for C = 1000. Thus, for the Gaussian SVM the hyperparameter value that results in the optimal performance of the algorithm (in this case recall and precision) is C = 100.

## *B. Comparison Analysis for Tuned Algorithmns*

*Table 2: Generalized ranking for all seven algorithms based on the computed evaluation metrics.*

| | Ranking (Highest = Left, Lowest = Right) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Random Forest | Gaus. SVM (C= 100) | Log. Reg L2, λ = 1 | Log. Reg No Regularization | Log. Reg L1, λ = 10 | Poly Order 2 Log. Regr. No Regularization | Poly SVM Degree 1 |
| Recall | 0.785 | 0.776 | 0.623 | 0.62 | 0.619 | 0.601 | 0.331 |
| Precision | 0.937 | 0.848 | 0.876 | 0.867 | 0.87 | 0.837 | 0.824 |
| F1 | 0.852 | 0.81 | 0.727 | 0.722 | 0.722 | 0.698 | 0.472 |
| Accuracy | 1 | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 |

A summary of the overall algorithm ranking has been shown in *Table 2* above. As previously mentioned, priority is given to the overall *recall* score of algorithms when being compared due to the high error costs to the bank/customer when false negative predictions occur. Analysing *Table 2* shows that the polynomial SVM with degree *1* had the lowest *recall* score of *0.331*, with the 2nd order polynomial feature logistic regression (no regularization) performing significantly better with an overall *recall* score of *0.601*.

Surprisingly, the logistic regression with L1-regularization performed slightly worse than the logistic regression with no regularization, with a recall score of *0.619* and *0.620* respectively. Logistic regression with L2-regularization performed marginally better with a *recall* score of *0.623*. It can be observed from these results that the logistic regression algorithms performed generally the same with small differences between *recall* score when various regularization approaches were taken.

The Gaussian SVM with regularization parameter *C* set to *100* performed noticeably better than the logistic regression algorithms with a *recall* score of *0.776*. Finally, the best performing algorithm in terms of *recall* score was the random forest algorithm with a *recall* score of *0.785*.

The ranking of algorithms follows a similar suit to *recall* score when the *F1-score* is evaluated and a slight shift in ranking occurs when the *precision* score is considered. *Accuracy* for all algorithms is high due to the data (train and test) being inherently biased towards non-fraudulent data, which consequently results in the algorithms being biased by this as well. Hence, being biased to classify test cases as negative and the high probability of negative cases causes a saturation of this metric and is only shown as reference.

Comparing Gaussian SVM to all variants of the logistic regression, the nonlinear SVM performed generally better. One possible explanation to the higher performance of this SVM could be the separability of the classes for the data. As the logistic regression for most variants uses a linear model, the nonlinearity of the Gaussian kernel could provide an advantage to the SVM (especially when comparing the linear SVM to Gaussian SVM). Interestingly, when a nonlinear model was used for the logistic regression the performance was worse, this may be in part due to some amount of overfitting and could be alleviated with regularization. It may be suspected that the SVMs nature to optimize margins when setting the decision boundary gives the algorithm an advantage as seen when comparing the nonlinear SVM (Gaussian SVM) to the nonlinear logistic regression ($2^{nd}$ degree polynomial logistic regression).

Of all the tested algorithms, random forest (default parameters, number of trees are *100*) produced the best results across the board. One potential reason for this increase in performance over even Gaussian SVM, is that the depth of the decision trees allows the random forest to produce decision boundaries that are more intricate than the Gaussian SVM. This allows the random forest to better separate classes in nonlinear problems. While having intricate decision boundaries exposes the algorithm to the risk of overfitting by having multiple sets of decision trees and aggregating the results into a final decision boundary, random forest can minimize the effects of overfitting that come with more intricate decision boundaries. As a result, random forest can perform well in these types of nonlinear classification problems.

## VI. Conclusions

Based on the results and analysis, of the tested algorithms the ideal credit card fraud detection model is with the random forest algorithm. Its high *recall* and *F1-score* provide the best performance in catching fraud cases while also ensuring the model is not too sensitive and flagging non-fraudulent cases as fraud. It is a reminder that this decision is accompanied with the fact that the model was trained and tested on a relatively biased dataset from 2013. Given the increase in online shopping and user data collection in 2021, the occurrence of credit card fraud may be more frequent and the performance of the algorithm within this paper may no longer be reflected in real-world cases. As is the case with most machine learning problems, a potential solution to this is to train the algorithm on newer/more diverse data.

One major lessoned learned from this project was how to account for the effects of the data on the trained model. The bias in the training data to favour non-fraudulent data made the trained algorithms biased as well. With the test data seeing the same distribution as the training data and naively using accuracy as the sole performance metric the trained algorithm would seem to behave excellently. However, realizing this bias and learning methods to account for it such as different performance metrics (*recall*, *precision*, *F1*) and cross validation allowed for the handling of this bias, and taught valuable tools that can be applied in future work.

Regarding future work, the results obtained from this project can serve as another point of reference to foundational concepts in machine learning and a basis for credit card fraud detection. Additionally, with the use of data from more recent time, this work can serve as a point for comparison not only in how credit card fraud has potentially changed, but how some classical machine learning methods fair in today's "always online" environment.

## VII. References


[1] A. Bloomenthal, "How Credit Cards Work," *Investopedia*, 09-Apr-2021. [Online]. Available: https://www.investopedia.com/terms/c/creditcard.asp. [Accessed: 10-Apr-2021].

[2] F. C. A. of Canada, "Government of Canada," Canada.ca, 12-Nov-2019. [Online]. Available: https://www.canada.ca/en/financial-consumer-agency/services/credit-fraud.html. [Accessed: 10-Apr-2021].

[3] S. Rate, "Credit Card Fraud Statistics In Canada 2021," *Simple Rate*, 08-Jan-2021. [Online]. Available: https://www.simplerate.ca/credit-card-fraudstatisticscanada/#:~:text=89%25%20of%20Canadians%20have%20reported,at%20least%20one%20credit%20card.&text=According%20to%20The%20Canadian%20Bankers,fraud%20has%20increased%20by%2071%25. [Accessed: 10-Apr-2021].

[4] S. S. -, "Credit Card Fraud Detection Techniques," *StaySafe.org*, 09-Jan-2018. [Online]. Available: https://staysafe.org/credit-card-fraud-detection-techniques/. [Accessed: 10-Apr-2021].

[5] J. Brownlee, "Imbalanced Classification with the Fraudulent Credit Card Transactions Dataset," *Machine Learning Mastery*, 20-Aug-2020. [Online]. Available: https://machinelearningmastery.com/imbalanced-classification-with-the-fraudulent-credit-card-transactions-dataset/. [Accessed: 11-Apr-2021].

[6] L. Frei, "Detecting Credit Card Fraud Using Machine Learning," *Medium*, 25-Jan-2019. [Online]. Available: https://towardsdatascience.com/detecting-credit-card-fraud-using-machine-learning-a3d83423d3b8. [Accessed: 11-Apr-2021].

[7] M. L. G.- ULB, "Credit Card Fraud Detection," *Kaggle*, 23-Mar-2018. [Online]. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud. [Accessed: 10-Apr-2021].

[8] G. Developers, "Classification | Machine Learning Crash Course | Google Developers," *Google*. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/video-lecture. [Accessed: 10-Apr-2021].