Credit Card Fraud Detection

Paulo Pereira (98430)
DETI
Universidade de Aveiro
Aveiro Portugal
paulogspereira@ua.pt

***Abstract*—Currently, every person and corporation of the world is susceptible to fraudulent transactions, so it's only natural to conclude that a system that can automatically detect such transactions is very much needed. With that in mind around 280000 transactions were collected and analyzed. Every transaction has 30 columns, although only 29 have data the last one refers to the validity of the transaction. The Vnumber notation was used to refer to the 27 columns that were looked at; the data set also appears to be unbalanced.**

***Index Terms*—Model, Logistic Regression,SVM,Neural Network,Confusion Matrix, F1 Score ,Features, Learning Curve**

I. Introduction

This report details the techniques, methodologies, and algorithms used in the first AA (Automated Learning) project at Universidade de Aveiro. The project involved applying machine learning techniques learned in class or self-taught to solve one of several problems proposed by the course instructor, Petia Georgieva.
I chose to tackle the problem of credit card fraud detection by using algorithms to accurately identify fraudulent and non-fraudulent transactions. The aim was to determine the best model and parameters to achieve optimal results. I found this topic compelling because it concerns a serious matter involving significant amounts of money, and a well-designed model could be useful for future work in electronic systems that require secure transactions, which is a field of study I find particularly interesting as well as setting a foundation for a line of work I'm very found of which is field of AI, as well as, touching machine learning that is the main focus of my Masters degree (Robotics and Inteligent Systems), which is a field that I wish to be proficient at.

II. State of the art

Before starting the work, I aimed to understand better what this dataset [1], what results had been found, and which models produced them, while doing this part of the research in Kaggle, I found a notebook [2], that did what I wanted and I looked over it's contents in order to understand a better approach as well as how things could be done. The overall architecture of this notebook was followed, which makes it the bigger foundation for my overall approach. Still other notebooks were looked at for further inspiration. I also created and implemented other methods in certain parts that made more sense to my train of thought. Despite the Neural Network model proved to be the best in this case, other models such as LogisticRegression, KNeighborsClassifier, SVC, DecisionTreeClassifier, and SMOTE were also explored.
The results obtained by the previously mentioned notebook [2] were as follows:

Model	Score(%)
Logistic Regression	95.0
KNeighborsClassifier	93.0
SVC	92.0
DecisionTreeClassifier	88.0
SMOTE	96.0

Table 1 - ACCRUACY RESULTS.

My main aim was to do better than these findings with the models I want to train, and asses not only the accuracy but also other outcomes that have not been reported, which I will cover later in this report, such as confusion matrices, f1 score, precision, prediction results and others
As previously mentioned, there were other notebooks that have been looked at, from these notebooks one notebook [3] contained similar works as the linked works referenced in previously. While the focus for this task was on Logistic Regression, which had been studied in AA courses, I will also refer to this notebook to make comparisons and draw conclusions. The notebook in notebook [3] places special emphasis on the recall value, and I will explain later why this value is important for the task. The recall value achieved in that notebook [3] was approximately 93%.

III. DATASET ANALYSIS

A. dataset description

The data [1] used for this project, presents a total of 284807 transactions, of which only 492 are frauds. The dataset consists of 31 columns for each transaction, including Time, V1 to V28, Amount, and Class. The first 30 columns are features, and many of them have names with the Vnumber syntax for security reasons. While some of these features should be text, they have been transformed into decimal numeric values. The last column, "Class," indicates whether the transaction was fraudulent or not, with "0" representing a normal transaction and "1" representing a fraudulent one.

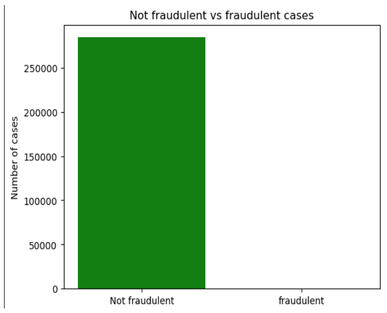

Figure 1 - Non Fraudulent vs Fraudulent cases

As mentioned earlier, the dataset contains far more non-fraudulent cases than fraudulent ones. The number of non-fraudulent cases is significantly higher, accounting for approximately 99.83% of the dataset.
This can create challenges in developing a classifier since it may mistakenly classify a fraudulent case as non-fraudulent due to the overwhelming number of non-fraudulent cases. In this work, however, it is preferable to classify a non-fraudulent transaction as fraudulent, which presents a unique challenge. Fig. 1 illustrates the distribution of non-fraudulent and fraudulent cases in the dataset.

B. Sub-Sample of the dataset

To tackle the mentioned problem I decided to sub-sample the dataset in a way that fraudulent cases and non-fraudulent cases were present in equal amount. This would prove to be enough to minimize the problem of the classifier misclassifying results. The following picture shows the subsampled version of the previous figure.

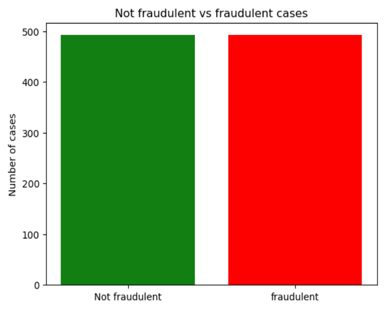

Figure 2 - Sub-sampled Non Fraudulent vs Fraudulent cases

To demonstrate that the dataset was well-balanced, i included a Naive Predictor [4], which is a simple classification model that assumes little or nothing about the problem. Its performance provides a baseline against which all other models evaluated on a dataset can be compared. My intention was to have the classifier predict all transactions as non-fraudulent to emphasize that the model lacks intelligence. This approach allowed me to confirm the expected results when there is no intelligence and provided a benchmark for comparison with my subsequent models.

The initial dataset yielded the following results:

- Accuracy score: 0.9983
- Precision: 0.9983
- Recall: 1.0000
- F1-score: 0.9991

The subsampled dataset produced the following results:

- Accuracy score: 0.5000
- Precision: 0.5000
- Recall: 1.0000
- F1-score: 0.6667

I anticipate that my models will yield higher accuracy, precision, and f1-score values than those obtained with the subsampled dataset. Since I selected non-fraudulent transactions at random, it should be noted that any future work with the classifiers will be based on the updated dataset.

C. Feature Normalization

At this point data still needed to be normalized due to some features have much higher values than others, this would skew the results and not accomplish the objective of each feature holding a similar weight. I used StandardScaler to normalize the data, by removing the mean and scaling to unit variance.

D. Feature analysis

I analyzed a dataset with a significant number of characteristics to determine whether the values of each feature in all transactions were comparable across fraudulent and non-fraudulent instances. The goal was to identify which attributes would provide greater evidence that a fraud has occurred. To achieve this, I created a bar graph for each feature and looked for features with more distinct values, as these could indicate greater relevance.

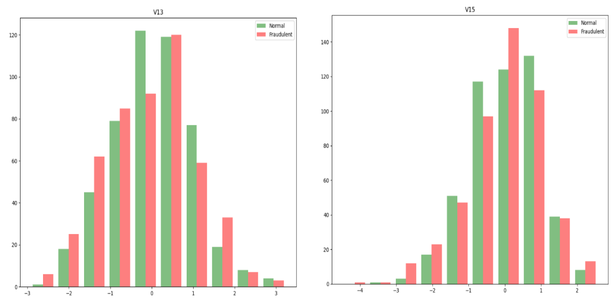

Figure 3 - Example where there is no difference between both cases (V13/V15)

By observing the bars of both cases in Fig. 3, I concluded that V13 and V15 should not greatly influence the detection of fraudulent transactions, as the bars in both cases were almost always the same for different values.

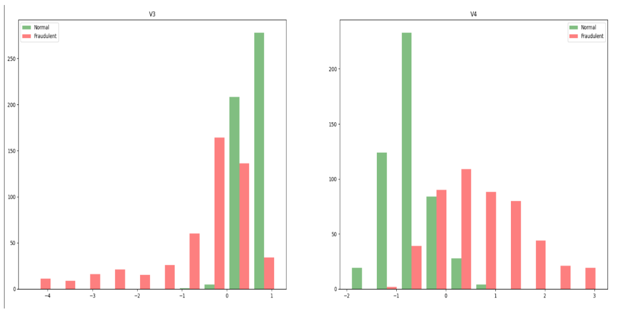

Figure 4 - Example where there is a big difference between both cases (V3/V4)

In contrast, Fig. 4 showed a big difference between the bars for V3 and V4, indicating that these characteristics are more likely to be altered when fraud is present. Therefore, evaluating these features in greater depth may yield better results than analyzing the set of 30 features. Based on my analysis, the attributes V2, V3, V4, V5, V10, V11, V12, V14, V16, V17, V18, V19 exhibit the greatest differences between situations of a transaction being fraudulent or not fraudulent.

I used the correlation matrix method to examine the data to determine which characteristics are typically more indicative of whether or not there is fraud. I predicted that the features completed as main indicators in this phase would be the same or nearly the same as those done in the previous phase. To demonstrate the importance of the sub-sample, I created this matrix for both the sub-sample dataset and the initial dataset.

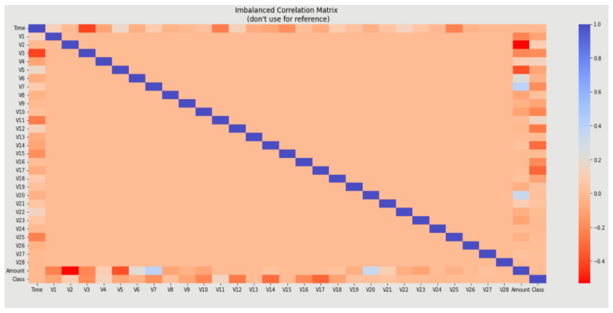

Figure 5 - Correlation matrix for initial dataset

Observing the correlation matrix of the original data in Fig. 5, I found it challenging to conclude which features have more weight in the Class line, which is where we can see the indicator of whether or not a transaction is fraudulent. In this situation, all values were very much in the range [-0.2,0], and what is wanted are the values of the extremes, that is, values closer to either -1 or 1.

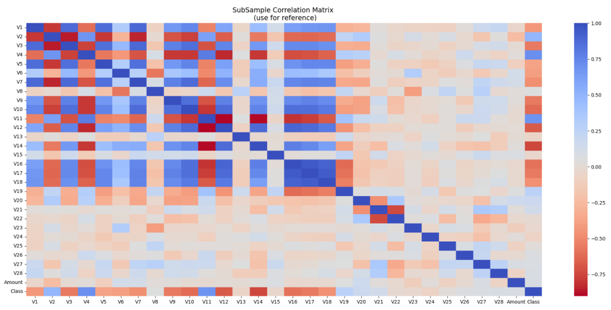

Figure 6 - Correlation matrix for sub-sampled data

However, when looking at the correlation matrix of the sub-sample data in Fig. 6, I could already see the values near -1 and 1 in the Class line that I was hoping to observe. From this, I could extrapolate that these characteristics are the ones that offer a larger indication of whether there is a fraud when they are in specific values. The features that are closer to blue have a positive correlation, indicating that the higher the value of this feature, the more likely it is to be a fraudulent transaction. These features are V4 and V11. The features that are closer to red have a negative correlation, indicating that the lower this value is, the more likely the transaction is to be fraudulent. These features are V3, V9, V10, V12, V14, V16 and V17.

To sum up, the best features to be used as indicators are: V3, V4, V9, V10, V11, V12, V14, V17.

To complete this section, I chose the approach of creating box plots for the most valuable features. [5] ”A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”)”. With this techinque, it is easier to find the outliers and see how symmetric and grouped together the data is.

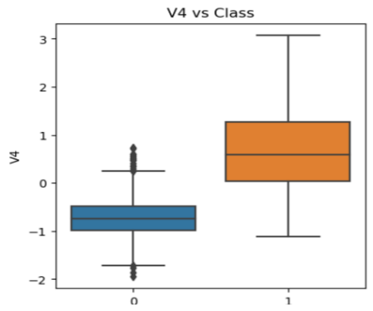

Figure 7 - Boxplot for V4 vs Class

According to the Figure above, we can see that a feature that was determined to have a positive correlation mean that it's box is below the Class box, this assumption is fair because this is the case for every feature with a positive relation. Not only that but the reverse was also true, every case with a negative relation would present a feature box above the Class box.

Outliers are just points that show outside (after) the horizontal lines. These values must be deleted and the boxplot readjusted. The features where this is more relevant are V3, V9 and V10. The following figure shows the boxplot after this removal was done.

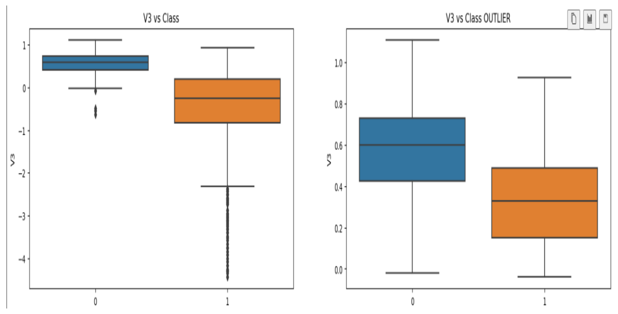

Figure 8 - Boxplot for V3 vs Class and for V2 OUTLIER vs Class

Since a result, these outcomes were much more concentrated in a shorter range, which would likely aid when training the models, as the ”irregular” examples were excluded, and an improvement in percentages is predicted.

IV. Machine Learning Models

A. Description of the implemented models

As training models, I've decided to go with three we've explored in class, which are: Logistic Regression, Naive Bayes and SVC.
The Logistic Regression algorithm [6] [7] is commonly used to classify observations into specific groups. It differs from Linear Regression, which generates continuous number values, by applying the logistic sigmoid function to transform its output into a probability value that can be assigned to multiple discrete classes. To optimize the model's performance, we can use the sklearn function and various parameters to train it. Initially, we can randomly select these parameters to verify the model's functionality, and then we can use hyperparameter [8] tuning to determine the best parameter values without having to manually test different combinations. This approach was also applied to the SVM and Neural Network models.

The Support Vector Classification algorithm [9], which is based on libsvm, is often used for classification tasks but can take a long time to train when there are a large number of samples. Because we had experience with it in class, we always utilized the rbf kernel parameter.

Neural Networks [10] are a type of machine learning model that follows a simple process for making predictions: inputting data, generating a prediction, and comparing it to the desired output. The model then modifies its weights and biases to minimize the difference between the predicted and desired outputs, effectively improving its performance over time.

B. Explanation of the outcomes for each model to be examined

1) Precision score: To calculate this metric, I used sklearn [11] and the for the precision score itself, the equation comes to
$$precision = tp/(tp + fp)$$
where tp is the number of true positives and fp the number of false positives. This means that precision is the system's ability to not label as positive a sample that is negative. And consequently it will tell us how many of the positive classifications are correct.
2) Recall score: To calculate this metric sklearn [12] was used once again. The recall score is calculated by:
$$recall = tp/(tp + fn)$$
where tp is the number of true positives and fn the number of false negative. Which means this metric represents the ability of the system to find all positive samples.
As a side note, it should be mentioned that both precision and recall fluctuate between 0 and 1, where 1 is the best value.
3) F1 score and accuracy score: To calculate this metric sklearn [13] was also used. we used [15] and for [15] f1 score is : The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. It's formula is:
$$F1 = 2 * (precision * recall)/(precision + recall)$$
the accuracy score represents the overall performance of the system.
4) Confusion Matrix: this is basically a table that indicates the successes and mistakes of the model, compared to the expected result.
To compute the matrix sklearn [14] will once again be used.
5) Learning curve graph: Training score is mostly applied in machine learning to estimate the skill of a model on seen data.
Cross-Validation is primarily used to estimate the skill of a model on unseen data.
Over-fitting is indicated by high training score and a poor test score. Under-fitting happens when both scores are low.
6) Scalability: This indicates how much execution time rises with the amount of training, which means that we should select the one with the shortest execution, for the same number of training and assuming scores are the same. This can be mostly used as a tiebreaker when two models have similar results.
7) Performance: Indicates how the execution affects the score, if two modes have the same results this metric can also be used as a tie breaker.

C. Model training

To train the aforementioned models, we first utilized the train_test_split [15] function from sklearn in an initial phase.

This function required input of an ”X” and a ”y”. It’s important to note that ”X” represents the features, while ”y” represents the ”Class” column. As an output of this function, we obtained the variables X_train, X_test, y_train, and y_test, which were subsequently used to generate the results.

D. *Model training results*

The results come from all features apart from ”Time” as well as from the scenario where the ”X” will only contain the best features mentioned previously.

1) Logistic Regression without penalty: In this one i kept mostly all the default settings from the functions. Only increasing the number of iterations to 5000 and setting the penalty to none.

Scores	All Features	Best Features
F1	0.9184	0.9593
Accuracy	0.9187	0.9593
Precision	0.9725	0.9829
Recall	0.8618	0.9350

Table 2 - LOGISTIC REGRESSION WITHOUT PENALTY.

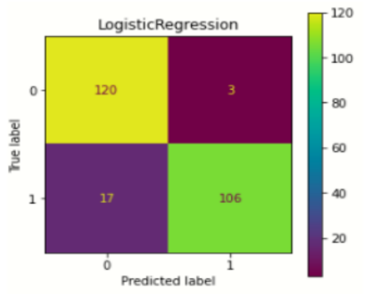

Figure 9 - Confusion Matrix for Logistic Regression(all features) without penalty

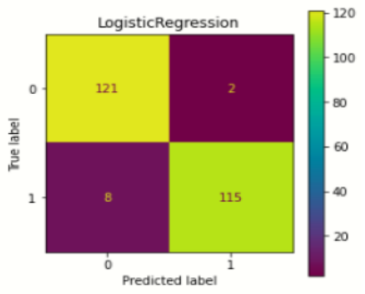

Figure 10 - Confusion Matrix for Logistic Regression(best features) without penalty

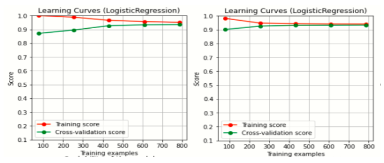

Figure 11 - Learning curve for Logistic Regression(all features vs best features) without penalty

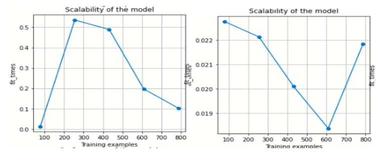

Figure 12 - Scalability of Logistic Regression(all features vs best features) without penalty

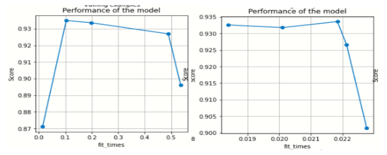

Figure 13 - Performance of Logistic Regression(all features vs best features) without penalty

Upon comparing the results of my Logistic Regression model for two scenarios that I wanted to evaluate, I have found that using just the best features yields better results than utilizing all features. This improvement is consistently evident across all values that I have evaluated. The effectiveness of the predictions is particularly noticeable in the confusion matrix, where the model with just the best features performed significantly better.
By analyzing the learning curves, I have determined that the optimal model for the best features can be achieved with only about 400 training cases.

2) Logistic Regression with penalty ”l2”: I used almost all of the logisticRegression’s default settings, increasing the iterations to 5000 and setting the penalty to ”l2” in the model.

Scores	All Features	Best Features
F1	0.9184	0.9471
Accuracy	0.9187	0.9472
Precision	0.9813	0.9825
Recall	0.8537	0.9106

Table 3 - LOGISTIC REGRESSION WITH PENALTY ”L2”.

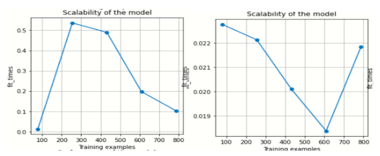
Figure 14 - Confusion Matrix for Logistic Regression(all features) with penalty ”L2”

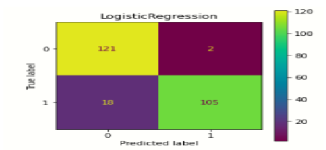
Figure 15 - Confusion Matrix for Logistic Regression(best features) with penalty ”L2”

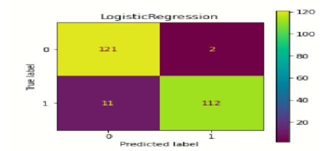
Figure 16 - Learning curve for Logistic Regression(all features vs best features) with penalty ”L2”

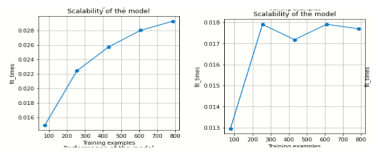
Figure 17 - Scalability of Logistic Regression(all features vs best features) with penalty ”L2”

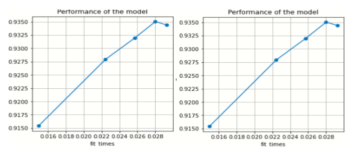
Figure 18 - Performance of Logistic Regression(all features vs best features) with penalty ”L2”

Similar to the previous situation, using the best features has resulted in improvements in the outcomes and confusion matrix in this case as well. While the behavior on the learning curve is nearly identical in both scenarios, there is something strange about the results when using the best features, and they don’t seem entirely correct.

3) SVC: For the SVC model, almost all the default settings were used once again, increasing the number of iterations to 5000 and the kernel to ”rbf”.

Scores	All Features	Best Features
F1	0.9143	0.9470
Accuracy	0.9146	0.9472
Precision	0.9722	0.9911
Recall	0.8537	0.9024

Table 4 - SVC Results.

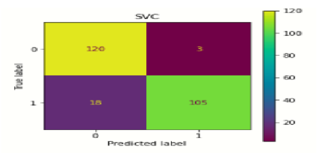
Figure 19 - CConfusion Matrix for SVC(all features)

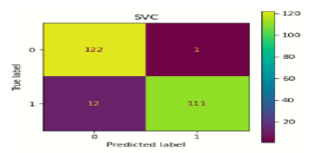
Figure 20 - Confusion Matrix for SVC(best features)

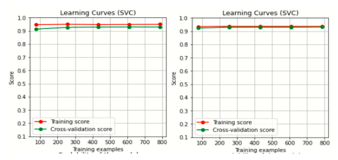
Figure 21 - Learning curve for SVC(all features vs best features)

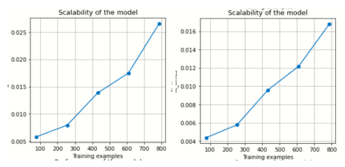
Figure 22 - Scalability of SVC(all features vs best features)

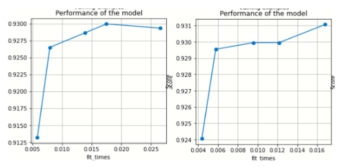
Figure 23 - Performance of SVC(all features vs best features)

Like in the previous model, there are improvements in both outcomes and confusion matrix. The learning curve looks unchanged, or at least very identical, in both cases. Despite that on case of the best features the results is weird and seams wrong.

4) NN: In this last model I used every setting on default besides the number iterations which was increased to 5000.

Scores	All Features	Best Features
F1	0.8861	0.9512
Accuracy	0.8862	0.9512
Precision	0.9060	0.9826
Recall	0.8618	0.9187

Table 4 - Neural Network

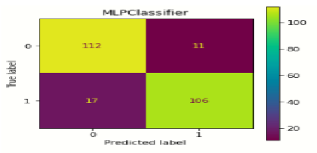

Figure 24 - Confusion Matrix for NN(all features)

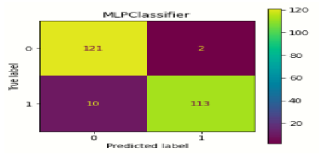

Figure 25 - Confusion Matrix for NN(best features)

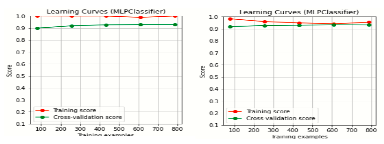

Figure 26 - Learning curve for NN(all features vs best features)

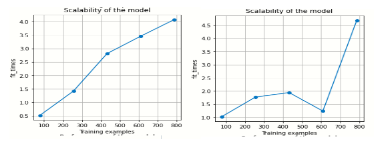

Figure 27 - Scalability of NN(all features vs best features)

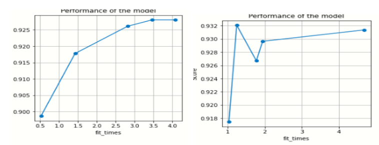

Figure 28 - Performance of NN(all features vs best features)

As in previous cases, there are visible improvements in the outcomes and confusion matrix. The learning curve looks unchanged, or at least very identical, in both cases. Despite that on case of the best features the results is weird and seams wrong.

E. Analysis of results

In this part we will compare the best case of each model; the following table display the contrast between models.

Models	F1 Score	Accuracy Score
Logistic Regression	0.9593	0.9593
Logistic Regression with ”L2”	0.9471	0.9472
SVC	0.9470	0.9472
NN	0.9512	0.9512

Table 5 - F1 SCORE AND ACCURACY SCORE VALUES.

Models	Precision Score	Recall Score
Logistic Regression	0.9829	0.9350
Logistic Regression with ”L2”	0.9825	0.9106
SVC	0.9911	0.9024
NN	0.9826	0.9187

Table 6 - PRECISION SCORE AND RECALL SCORE VALUES.

Models	False Positive	False Negative
Logistic Regression	2	8
Logistic Regression with ”L2”	2	11
SVC	1	12
NN	2	10

Table 7 - FALSE POSITIVES AND FALSE NEGATIVES VALUES.

Models	FP+FN	Fit time rank
Logistic Regression	10	3º
Logistic Regression with ”L2”	13	2º
SVC	13	1º
NN	12	4º

Table 8 - FP+FN AND FIT TIME RANK.

When analyzing the results obtained from the various models, I will exclude comparisons with ”L2” because their results are consistently worse than those of Logistic Regression without penalty, except for fit time. However, since the fit time for both models is around 0.0x, we can conclude that their performances are similar. In this initial phase, I will compare SVC, NN, and Logistic Regression without penalty to determine the best model based on the data obtained, without hyper tuning of the parameters.

All of the models presented have excellent values, but there are differences, and it is these differences in conditions that will lead to a high-quality model in these initiatives. Logistic Regression has the best results in both fields shown in Table 3, followed by NN and SVC, while Logistic Regression still has slightly higher numbers than the others.

Table 4’s results will not be used because Precision Score and Recall Score values cause False Positive and False Negative results. Therefore, in this work, false negatives are more critical because they mean that the transaction is fraudulent, but the classifier identified it as non-fraudulent. In contrast, if the classifier identifies a non-fraudulent transaction as fraudulent, it is not as critical. Based on the fewest False Negatives and wrong values (10) observed in the FP+FN Column of Table 6, its safe to conclude that Logistic Regression provides the best results.

Therefore, based on these initial values, I conclude that Logistic Regression offers the best results and provides its best model near the 400 training samples at the learning curve level.

V. GRADIENT BOOSTING REGRESSOR

While searching for the best possible result, I came across the Gradient Boosting Regressor [16], which allows for optimization of the arbitrary loss functions. It will be used to determine the most important features, it’s results are better than those previously achieved.

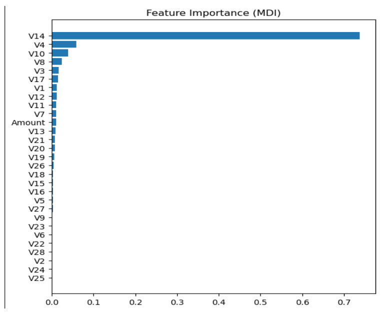

Figure 29 - Feature Importance by Gradient Boosting Regressor

The results obtained from the previous figure were analyzed in two situations: using only feature 14 and using the five features identified as the most important. However, the results were not the best, so only the scores are presented in this report.

Table 7 shows the results for the Gradient Boosting Regressor using the two feature selection situations. As can be seen, the results are worse than expected, but i still found it important and interesting to present these results. Although the selected features were considered the most important by the method, the results were not as expected, with lower scores compared to the previous results.

Models	FP+FN	Fit time rank
F1	0.8980	0.9185
Accuracy score	0.8984	0.9187
Precision	0.9537	0.9640
Recall	0.8374	0.8699

Table 9 - GRADIENT BOOSTING REGRESSOR RESULTS

VI. TUNING HYPER-PARAMETER

In the ML Model section, it was observed that default values were used for most of the model parameters. However, it is not guaranteed that these settings would produce the best possible results for the models.

To avoid the exhaustive task of manually searching for the best parameters for each model, it was decided to perform parameter tuning. This involves defining all possible values of the parameters and testing all possible combinations using the computer. The best combination of parameters will then be determined, and the models will be tested again with these optimal parameters to check if better results can be achieved.

A. Logistic Regression Parameters

The following table will be used to determine the values the following table will be used to determine which parameters are suggested for Logistic Regression:

Solver	Max iterations	class weight	penalty	C
lbgfs	5000	balanced	”none”,”l2”	[0.001, 0.01, 0.1, 1, 10,100, 1000]
liblinear	5000	balanced	”l1”,”l2”	[0.001, 0.01, 0.1,1,10, 100,1000

TABLE I
TABLE 10 - LOGISTIC REGRESSION PARAMETERS ALL POSSIBLE VALUES

The classification report yielded the following

Solver	Max iterations	class weight	penalty	C
lbgfs	5000	balanced	12	1

TABLE II
TABLE 11 - LOGISTIC REGRESSION BEST PARAMETERS

	precision	recall	f1 score	support
0	0.94	0.97	0.95	123
1	0.97	0.93	0.95	123
accuracy			0.95	246
macro avg	0.95	0.95	0.95	246
weighted avg	0.95	0.95	0.95	246

TABLE III
TABLE 12 - LOGISTIC REGRESSION CLASSIFICATION REPORT

B. SVC Parameters

The following table will be used to determine the values the following table will be used to determine which parameters are suggested for SVC:

kernel	C	gamma
rbf	[0.001,0.01,0.1,1,3,10,30,100,1000]	[0.001,0.01,0.1,1]

TABLE IV
TABLE 13 - SVC PARAMETERS ALL POSSIBLE VALUES

The classification report yielded the following:

kernel	C	gamma
rbf	3	1

TABLE V
TABLE 14 - SVC BEST PARAMETERS

	precision	recall	f1 score	support
0	0.94	0.97	0.95	123
1	0.97	0.93	0.95	123
accuracy			0.95	246
macro avg	0.95	0.95	0.95	246
weighted avg	0.95	0.95	0.95	246

TABLE VI
TABLE 15 - SVC CLASSIFICATION REPORT

C. NN Parameters

The following table will be used to determine the values the following table will be used to determine which parameters are suggested for NN:

solver	max iterations	hidden layers size	activations	alpha	learning rate	rate init
adam	5000	(12,12), (12,12,12)	tanh,relu	1e-3,1e-4	constant,invscaling	0.001,0.01

TABLE VII
TABLE 16 - NN PARAMETERS ALL POSSIBLE VALUES

The classification report yielded the following

solver	max iterations	hidden layers size	activations	alpha	learning rate	learning rate init
adam	5000	(12,12)	tanh	1e-3	constant	0.001

TABLE VIII
TABLE 17 - NN PARAMETERS BEST VALUES

	precision	recall	f1 score	support
0	0.92	0.98	0.95	123
1	0.98	0.92	0.95	123
accuracy			0.95	246
macro avg	0.95	0.95	0.95	246
weighted avg	0.95	0.95	0.95	246

TABLE IX
TABLE 18 - NN CLASSIFICATION REPORT

D. Results analysis

After obtaining the previous results for hyper-parameter tuning, I repeated the confusion matrix and the graph of the learning curves to ensure that the values were as good as they appeared to be in the classification report of each model. The matrix, in particular, made it easier for me to compare the results that had been obtained.

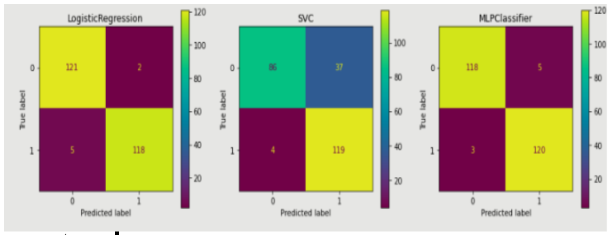

Figure 30 - Confusion Matrix for all modules

As I observed from the findings, there were considerable gains in both Logistic Regression and MLPClassifier (NN), while the SVC performed much worse in terms of false positives while improving in false negatives. I did not comprehend what happened in the SVC to cause the results to worsen; nevertheless, because the key goal is that there are no false negatives and this number has decreased, I continue to believe that the SVC result was not that awful.

Now, analyzing the results in detail, and using Fig. 35, it can be seen that the Neural Network is the best model at the moment, as the main emphasis is now given to false negatives and the NN number is only 3, which translates into a very high recall, around 0.9750, and in this case, the Neural Network is the best model. As previously stated, recall is one of the main metrics to be examined; however, it still produced 5 false positives, implying that Logistic Regression was the best model because this sum is only 7 while NN is 8, but it is considered more critical that the model produces 5 false negatives and 2 false positives rather than 5 false positives and 3 false negatives.

Having stated that, at this stage in my study, I have discovered that the best solutions are all features with high parameter tuning for Logistic Regression and NN, and optimal features without the parameters returned by high parameter tuning for SVC. So I simply opted to include the learning curve for the situations where there were genuine improvements in the outcomes, as well as the tables containing all of the data I acquired in this part.

Model	precision score	recall score	f1 score	accuracy score
Logistic Re-gres-sion	0.9833	0.9593	0.9715	0.9715
1	0.9600	0.9750	0.9675	0.9675

TABLE X
TABLE 19 - RESULTS FOR LOGISTIC REGRESSION AND NN

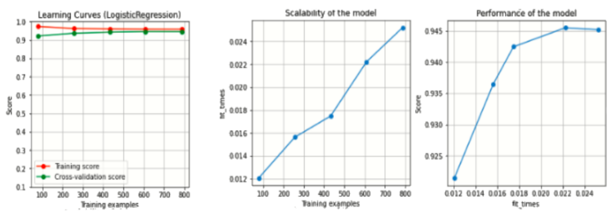

Figure 31 - Learning Curve for Logistic Regression

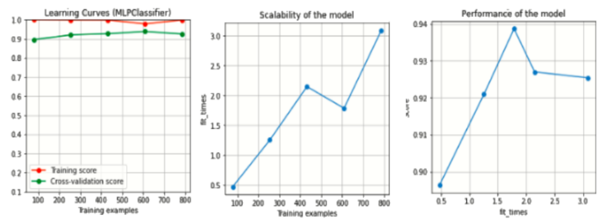

Figure 32 - Learning Curve for NN

Based on the results, I observed that while the Neural Network had a better recall score in the confusion matrix, the remaining values were consistently worse than the Logistic Regression. I also considered the performance and scalability of both models, and found that Logistic Regression is better and much more scalable in systems where high performance is required. The NN time is much higher than the Logistic one, which further reinforces the superiority of Logistic Regression in systems that demand high responsibility.

VII. K-FOLD

In order to further test the dataset, I decided to use the K-fold function [17] provided by sklearn. This approach differs from the basic train and test algorithm used in the initial phase of the work, and may lead to improved results. However, I only used the parameters that had previously been identified as the best, rather than all attempts made.

The K-fold function has several advantages, including providing an idea of how the model will perform on an unknown dataset and helping to determine a more accurate estimate of model prediction performance.
However, it is important to note that using this method requires repeating the procedure multiple times, making it more computationally intensive than a simple training.

Model	Average Accuracy(%)
Logistic Regression	92.81
NN	93.76
SVC	92.81

VIII. VALIDATION OF PARAMETERS

In order to prove that shown values are in fact the best, I used validation curves [?] and created some validation curves for the most important parameters in each model.

A. *Logistic Regression*

In the case of this model, I will only look at the C parameter. Despite the predicted one may not be the ideal value for this metric, but since it is a very close value to 1 the prediction winds up to be correct.

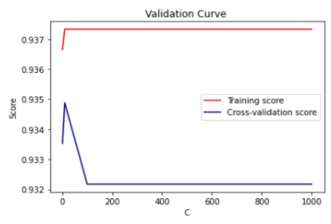

Figure 33 - Validation Curve for C

B. *SVC*

This time two parameters will be analyzed, gamma and C. As shown in the following figure we can see that the gamma value more or less stabilizes eventually, so the value of 1 looks reasonable.

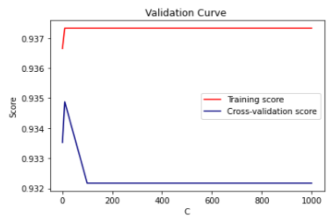

Figure 34 - Validation Curve for gamma

As shown now the value of 3 to the C parameter was a very good decision.

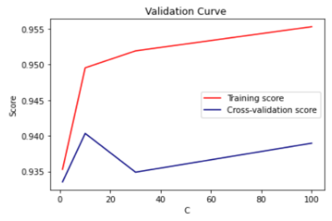

Figure 35 - Validation Curve for C

C. NN

Now let's take a look at two more parameters, learning rate init and alpha, as seen in the picture's the value chosen by the hyper-parameter model, 0.001, also corresponds to the best results provided from the validation.

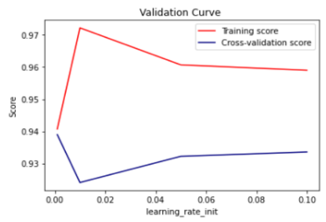

Figure 36 - Validation Curve for learning rate init

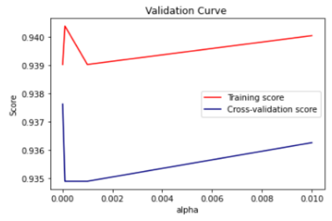

Figure 37 - Validation Curve for alpha

IX. Final Results

At this point let's remind ourselves of the best values for each model

For Logistic Regression:

- Solver: lbgfs
- Max iteratons: 5000
- class weight: balanced
- penalty: 12
- C: 1

For SVC:

- kernel: rbf
- C: 3
- gamma: 1

For NN:

- Solver: adam
- max iterations: 5000
- hiden layers size: (12,12)
- activations: tanh
- alpha: 1e-3
- learning rate: constant
- learning rate init: 0.001

The plan is to utilize the trained models by applying them to the entire initial dataset, and use the predictions generated by each model to determine the most effective one for detecting fraud.
In the following table and figure we can see the confusion matrix for all the values as well as the comparison between all models.

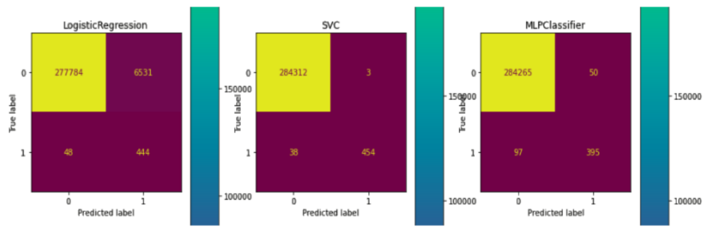

Figure 38 - Confusion Matrix for all values

Model	f1 score	accuracy score	precision score	recall score
Logistic Regression	0.9868	0.9769	0.0637	0.9024
SVC	0.9999	0.9999	0.9934	0.9228
NN	0.9995	0.9995	0.8876	0.8028

TABLE XI
Table 20 - Final Results

After analyzing the values of Table XXV and Fig 43, I have come to the conclusion that the best model for fraud detection is surprisingly the SVC. In all values, it presented itself as the one with the best results, with about 38 false negatives and only 3 false positives. Although this is still a high number, I was able to obtain a recall of around 92.28 percent, which is very positive, and the precision was also quite high, around 99.34 percent.
I was surprised to find that LogisticRegression provided such a low precision of only 6 percent despite its high recall of about 90 percent. The number of false positives was an outrageous 6531, suggesting that the model's training may not have been the most correct.
The Neural Network showed respectable results, with an accuracy and recall of around 80 percent. However, given

its performance and efficiency, which is considerably behind the SVC, nothing makes me want to use this model over the SVC.
Based on my analysis, I have concluded that the SVC is effectively the best model for fraud detection, and if I were to move to a real case, this would be the model to use. As seen in previous sections, this model is also the one that, at the level of efficiency, was presented as faster and with high performance.

X. COMPARING RESULTS WITH STATE OF THE ART

Comparing my results to two other Kaggle notebooks, I found some big differences. Both of them found their best results using the LogisticRegression classifier, even though they did not try SVC like I did. However, if we only look at the results, there are no major differences. My best recall score was 92.3%, joparga3's [3] notebook reached the 93.2% mark, and Janio Martinez Bachmann's notebook reached 94%. I found that Janio Martinez Bachmann's notebook [2] went beyond by comparing with the SMOTE technique, which I did not test. Instead of selecting only a few non-fraudulent cases to have a balanced dataset, SMOTE can generate synthetic fraudulent cases, allowing us to train the model with a much bigger sample. With that, he achieved an impressive 99.98% accuracy score which I was also able to get, but only by testing with the whole dataset. My undersampled dataset could only score 91.46%.

XI. CONCLUSION

This is my first work of this sort, which allowed me to grow in comfort in the machine learning sector.
I belive the results ended up being quite positive as well.

XII. REFERENCES

REFERENCES

[1] dataset used [Online] https://www.kaggle.com/datasets/mlg-ulb/ creditcardfraud.

[2] Notebook used [Online] https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets/notebook.

[3] Notebook used to compare results [Online] https://www.kaggle.com/code/joparga3/in-depth-skewed-data-classif-93-recall-acc-now/notebook.

[4] How to Develop and Evaluate Naive Classifier Strategies Using Probability [Online] https://machinelearningmastery.com/how-to-develop-and-evaluate-naive-classifier-strategies-using-probability/

[5] Boxplot [Online] https://pt.wikipedia.org/wiki/Diagrama de caixa/

[6] Logistic Regression [Online] https://scikit-learn.org/stable/modules/ generated/sklearn.linear model.LogisticRegression.html/

[7] Logistic Regression Definition [Online] https://ml-cheatsheet. readthedocs.io/en/latest/logistic regression.html

[8] Hyper-parameter tuning [Online] https://scikit-learn.org/stable/modules/grid search.html

[9] SVC [Online] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.htm

[10] Neural Network [Online] https://scikit-learn.org/stable/modules/neural networks supervised.html

[11] Precision score function from sklearn [Online] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision score.html

[12] Recall score function from sklearn [Online] https://scikit-learn.org/ stable/modules/generated/sklearn.metrics.recall score.html

[13] F1 score function from sklearn [Online] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1 score.html

[14] Confusion matrix function from sklearn [Online] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion matrix.htm

[15] Train Test function from sklearn [Online] https://scikit-learn.org/stable/modules/generated/sklearn.model selection.train test split.html

[16] Gradient Boosting Regressor [Online] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

[17] K-fold [Online] https://scikit-learn.org/stable/modules/cross validation.html

[18] Validation curve explanation [Online] https://www.geeksforgeeks.org/validation-curve/

[19]