IB Computer Science Extended Essay

Session: May 2022

Research Topic: Comparison of Logistic Regression and Support Vector Machine algorithms in fraud detection.

Research Question: "How does the efficiency of a Logistic Regression algorithm compare to that of a Support Vector Machine algorithm in terms of detecting credit card fraud?"

Wordcount – 3971

Contents

1. INTRODUCTION 4

2. BACKGROUND INFORMATION 5

2.1 CLASSIFICATION 5

2.2 LOGISTIC REGRESSION 6

2.3 SUPPORT VECTOR MACHINE 7

2.4 PERFORMANCE MEASURES 8

2.4.1 Confusion Matrix 8

2.4.2 Precision and Recall Scores 9

2.4.3 F-1 Score 10

2.4.4 Accuracy Score 11

2.4.5 Use of Performance Measures 11

3. DESCRIPTION OF DATASET 12

4. EXPERIMENTAL STUDY 14

4.1 AIM OF THE EXPERIMENT 14

4.2 SETUP AND MATERIALS 14

4.3 PROCEDURE 15

4.3.1 Preparation and Verification of Dataset 15

4.3.2 Splitting Dataset into Testing and Training 18

4.3.3 Training and Testing through Logistic Regression Algorithm 18

4.3.4 Training and Testing through Support Vector Machine Algorithm 19

4.4 HYPOTHESIS 20

4.5 RESULTS 21

4.5.1 Results of Logistic Regression 21

4.5.2 Results of Support Vector Machine 22

5. DISCUSSION AND EVALUATION OF RESULTS 24

5.1 DISCUSSION 24

5.2 EVALUATION 25

6. CONCLUSION 27

7. BIBLIOGRAPHY 28

8. APPENDICES 30

APPENDIX A – CODE IMPLEMENTED FOR EXPERIMENT 30

1. Introduction

Since the beginning of digital transactions, there have been various new methods to gain unauthorized access to someone else's finances. Thanks to rapid technological improvements, payments can now be performed by simply typing one's credit card information and pin code on a device, or by utilizing their biometric identification. As the usage of these digital payments continue to grow, so does the possibility for fraud.

Computer algorithms can be utilized to help in the detection and control of these frauds. "Machine learning (ML), a study of computer algorithms that improve automatically through experience and the use of data can help with this problem."[1]

Credit Card Fraud Detection using ML is a process that involves data analysis and the development of an efficient model that will help identify and prevent fraudulent transactions. Ideally, a training data set will be created that will allow algorithms to detect the specific characteristics that make a transaction more or less likely to be fraudulent. We are not only able to identify fraudulent transactions, but we are also able to identify them with high accuracy out of millions of transactions that occur every day, thanks to the disposal of powerful ML algorithms.

[1] Mitchell, T. M. (1997). *Machine Learning* (1st ed.). McGraw-Hill Education.

Two such highly efficient and popular algorithms which are used to help identify these transactions are the Logistic Regression and Support Vector Machine algorithm. Thus, raises the question which will be extensively explored in this essay; which of the two algorithms is more efficient and effective in properly detecting fraud. Through finding this we will be able to understand which algorithm is better to help solve the specific problem of fraud detection. Through an experiment the algorithms can be tested, evaluated and compared to help answer the focus of the essay.

2. Background Information

2.1 Classification

Classification is the process of predicting the classes of elements from given data points. It is a form of supervised learning where a sufficiently represented training dataset is used by a model to map out to which of the classes an element from a dataset would be part of. Credit card fraud detection is a type of classification problem called binary classification. Statistical binary classification is the task of categorizing elements in a data set into **two predefined** class labels; normal and abnormal. To help solve this task and classify the elements from a dataset to appropriate class labels, we make use of ML algorithms which efficiently do so.

2.2 Logistic Regression

The logistic regression algorithm is a powerful supervised ML algorithm that is widely used in binary classification problems and produces results by using independent variables. [2]It assesses the results using a dichotomous variable and is specifically used in binary classification because it produces only two possible outcomes. The algorithm identifies the best fit between the dependent variable and several independent variables. It is very popular in classification problems as it quantifies the elements that lead to categorization. Logistic Regression uses the sigmoid (logistic) function to find the relationship between variables. The sigmoid function is an S-shaped curve that can take any number with real value and map it between 0 and 1 but never on those limits.[3]

In terms of binary classification and specifically detecting credit card fraud, logistic regression is one of the first algorithms that is preferred due to how easy the algorithm is to implement and interpret. The algorithm is also very efficient to train; thus, it will be faster in processing and learning from big datasets with several elements like the one we will be dealing with. The logistic regression algorithm is also less inclined to overfitting with low dimensional datasets (like the one which will be used). This means that the algorithm won't wrongly recall a transaction as fraudulent. In terms of fraud detection, this aspect of the algorithm is both beneficial and detrimental. It will be helpful in the sense that most of the transactions that it does class as fraudulent will be correct and accurate, but it also means that the algorithm will be reluctant and struggle a bit to correctly classify elements in a dataset that doesn't have polar features.

[2] https://towardsdatascience.com/11-most-common-machine-learning-algorithms-explained-in-a-nutshell-cc6e98df93be
[3] https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16#:~:text=Difference%20between%20SVM%20and%20Logistic%20R

2.3 Support Vector Machine

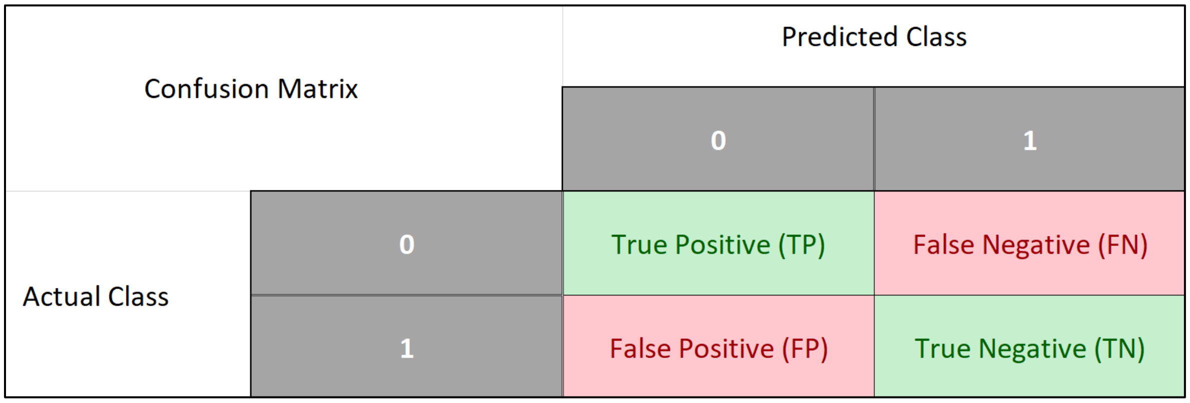
The support vector machine algorithm is a supervised ML algorithm that analyzes data for classification. The goal of support vector machine algorithms, which are based on statistical learning frameworks, is to identify a hyperplane in an N-dimensional space that can help classify data points into two categories. The support vector machine algorithm focuses on detecting a hyperplane with the maximum margin between the data points in the two classes and the dimensions of the hyperplane depends on the number of features in a dataset. This aspect of the algorithm aids in the classification of future data points with greater certainty.[4]

Like the logistic regression algorithm, the support vector machine is one of the first algorithms that is used when solving binary classification problems. The support vector machine however is preferred due to its extremely low computational power. One of the features of support vector machine is its regularization. This equips the algorithm with good generalization capabilities, stopping it from over-fitting. As mentioned for logistic regression, this feature of the algorithm is highly preferred for fraud detection due to its accurate classification. Another feature of the support vector machine algorithm is the stability of the model. This helps to maintain its results, which won’t be affected by small changes to the data. For imbalanced datasets like the one which will be used, stability of the algorithm will make it more probable and efficient to use.

[4] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

2.4 Performance Measures

2.4.1 Confusion Matrix



Confusion Matrix		Predicted Class	
		0	1
Actual Class	0	True Positive (TP)	False Negative (FN)
	1	False Positive (FP)	True Negative (TN)

Figure 1 – Visual Representation of a confusion matrix

A confusion matrix is a table that allows the performance of a classification algorithm to be shown. [5]It summarizes the model's correct and incorrect predictions using count values that are split down by class.

The matrix outputs four important values:

True Positive (TP) - values that are normal and accurately predicted as normal.

True Negative (TN) - values that are abnormal and are predicted abnormal.

False Positive (FP) - values that are abnormal but predicted normal by the algorithm.

False Negative (FN) – values that are normal but are predicted as false.

The matrix gives us information about the errors and what type of errors the algorithm has made. This feature helps us understand how the ML algorithm has classified the elements from a dataset,

[5] https://ijstr.org/final-print/dec2021/Utilizing-Data-driven-Approaches-To-Model-The-Risk-Of-Excavation-Damage-To-Underground-Natural-Gas-Facilities.pdf

where and how many errors the algorithm has made in classifying transactions so that the algorithm can be improved in the future. The most important use of the confusion matrix is through the four output values of it which will aid us in calculating performance scores, which will allow us to assess and investigate each algorithm's efficiency.

2.4.2 Precision and Recall Scores

Precision is the fraction of relevant instances among all the retrieved instances. [6]For detection of fraud, the formula of precision is $\frac{TN}{TN+\boldsymbol{FN}}$

Recall on the other hand is the fraction of retrieved instances among all the relevant instances and its formula for detection of fraud is $\frac{TN}{TN+\boldsymbol{FP}}$

High precision for an algorithm in fraud detection means that the algorithm is more likely to classify relevant results to a class label rather than the irrelevant ones. This score states the percentage of **correctly classified** transactions out of all the transactions the algorithm has labelled fraudulent. On the other hand, high recall for an algorithm in fraud detection means that the algorithm more likely returns most of the relevant results regardless of whether or not it returns the irrelevant ones. This score states what percentage of all the transactions that are actually fraudulent that the algorithm too correctly identifies as fraudulent.

[6] Somasundaram, A., Reddy, S. Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance. *Neural Comput & Applic* **31**, 3–14 (2019). https://doi.org/10.1007/s00521-018-3633-8

Other than the information these scores alone give us, they are used to generate the F-1 score using these ratios, which is a deterrent factor when evaluating the algorithms in binary classification problems.

2.4.3 F-1 Score

F-1 score is the weighted average of both precision and recall. It ranges from 0.0 to 1.0, where 1.0 is the highest possible value indicating a **perfect precision and recall** and 0.0 is the lowest value; either precision or recall is zero. We can calculate the F-1 score through precision and recall and the formula is $2 \times \frac{precision \times recall}{precision + recall}$.

Although the precision and recall scores will be analyzed to help decide which model is better, but more value should be placed on the F1 score as it combines the precision and recall metrics into one. The F1 score will play a vital role in comparing the ML algorithms as F1 score pays more importance to the false positive and false negative values. This is beneficial with the current dataset being used as it is heavily imbalanced. This means that the F1 score focuses on the minority class label, which is the abnormal class and it is important as the focus will be on the fraudulent transactions. This feature of F1 scores will give a highly specific scores on how efficient the ML algorithm is in identifying and classifying fraudulent transactions, which will help compare the models accurately.

2.4.4 Accuracy Score

An accuracy score is a representation of the correctly classified data from a dataset. The formula for this is $\frac{TP + TN}{Total\ number\ of\ values}$. Accuracy score, unlike the F-1 score, gives attention to the True positive and negative values. This means that the accuracy score will focus on how many fraudulent transactions and non-fraudulent transactions have been correctly identified. This means that with an imbalanced dataset, the accuracy score might not give us varied differences to compare the two models, but it is important for analyzing if the algorithm is accurate as a whole. This is good to know as we can calculate what percentage of the elements did the algorithm misclassify and work to improve it.

2.4.5 Use of Performance Measures

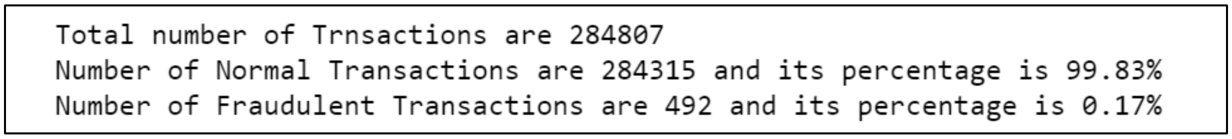
Now that we have a good understanding of what the metrics of comparison are, we need to look at how they will be used. While comparing the two ML algorithms, all 5 of the performance metrics (Confusion Matrix, Precision, Recall, Accuracy, F1 score) will be used to look at different angles of the models and see in what aspects do the algorithms excel at and where they can improve on. Through considering all the metrics we will be able to come to a final consensus.

The confusion matrix will be used to observe the work of the algorithms and calculating the performance scores. Precision and Recall will look at the behaviour and efficiency of the algorithms when dealing with fraudulent cases and also to calculate the F1 score. Accuracy score on the other hand will focus more on checking the similarities between the two algorithms to see if they are accurate algorithms in correctly identifying the classes of elements and not focusing on the model's ability to efficiently and accurately deal with fraudulent cases.

Lastly the F1 score; this metric is very important as it will play a defining role in deciding which algorithm is more efficient in identifying the fraudulent transactions as the score focuses on the minority class, which is the abnormal class of fraudulent cases.

3. Description of Dataset

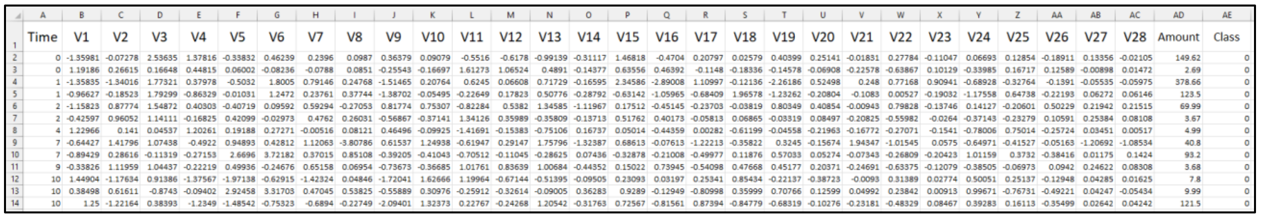
Now that the fraud detection classification problem and explanation of the focused ML algorithm have been discussed we shall move on to describe the features and characteristics of the dataset being used to evaluate the efficiencies of the two ML algorithms.



```
Total number of Trnsactions are 284807
Number of Normal Transactions are 284315 and its percentage is 99.83%
Number of Fraudulent Transactions are 492 and its percentage is 0.17%
```

Figure 2 – Statistics about the transactions in the dataset

The dataset contains transactions that have been made by European cardholders during **two days** in September 2012. From the image above we can see that there are a total of 284,807 transactions that have occurred and from those 492, or 0.17% of the transactions have been fraudulent and the rest of the 99.83%, or 284315 transactions have been valid and normal transactions. From this we can see that the dataset is highly imbalanced; representing one class (normal) much more than the other class (abnormal). I have chosen to use this dataset as it is highly representative of what happens in the real world where there are significantly more normal transactions than fraudulent ones. This will aid in finding the efficiency of these algorithms when used to deal with future scenarios that will continue to be imbalanced.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
1	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
2	0	-1.35981	-0.07278	2.53635	1.37816	-0.33832	0.46239	0.2396	0.0987	0.36379	0.09079	-0.5516	-0.6178	-0.99139	-0.31117	1.46818	-0.4704	0.20797	0.02579	0.40399	0.25141	-0.01831	0.27784	-0.11047	0.06693	0.12854	-0.18911	0.13356	-0.02105	149.62	0
3	0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.08236	-0.0788	0.0851	-0.25543	-0.16697	1.61273	1.06524	0.4891	-0.14377	0.63556	0.46392	-0.1148	-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.10129	-0.33985	0.16717	0.12589	-0.00898	0.01472	2.69	0
4	1	-1.35835	-1.34016	1.77321	0.37978	-0.5032	1.8005	0.79146	0.24768	-1.51465	0.20764	0.6245	0.06608	0.71729	-0.16595	2.34586	-2.89008	1.10997	-0.12136	-2.26186	0.52498	0.248	0.77168	0.90941	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
5	1	-0.96627	-0.18523	1.79299	-0.86329	-0.01031	1.2472	0.23761	0.37744	-1.38702	-0.05495	-0.22649	0.17823	0.50776	-0.28792	-0.63142	-1.05965	-0.68409	1.96578	-1.23262	-0.20804	-0.1083	0.00527	-0.19032	-1.17558	0.64738	-0.22193	0.06272	0.06146	123.5	0
6	2	-1.15823	0.87774	1.54872	0.40303	-0.40719	0.09592	0.59294	-0.27053	0.81774	0.75307	-0.82284	0.5382	1.34585	-1.11967	0.17512	-0.45145	-0.23703	-0.03819	0.80349	0.40854	-0.00943	0.79828	-0.13746	0.14127	-0.20601	0.50229	0.21942	0.21515	69.99	0
7	2	-0.42597	0.96052	1.14111	-0.16825	0.42099	-0.02973	0.4762	0.26031	-0.56867	-0.37141	1.34126	0.35989	-0.35809	-0.13713	0.51762	0.40173	-0.05813	0.06865	-0.03319	0.08497	-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.10591	0.25384	0.08108	3.67	0
8	4	1.22966	0.141	0.04537	1.20261	0.19188	0.27271	-0.00516	0.08121	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.16737	0.05014	-0.44359	0.00282	-0.61199	-0.04558	-0.21963	-0.16772	-0.27071	-0.1541	-0.78006	0.75014	-0.25724	0.03451	0.00517	4.99	0
9	7	-0.64427	1.41796	1.07438	-0.4922	0.94893	0.42812	1.12063	-3.80786	0.61537	1.24938	-0.61947	0.29147	1.75796	-1.32387	0.68613	-0.07613	-1.22213	-0.35822	0.3245	-0.15674	1.94347	-1.01545	0.0575	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	0
10	7	-0.89429	0.28616	-0.11319	-0.27153	2.6696	3.72182	0.37015	0.85108	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.07436	-0.32878	-0.21008	-0.49977	0.11876	0.57033	0.05274	-0.07343	-0.26809	-0.20423	1.01159	0.3732	-0.38416	0.01175	0.1424	93.2	0
11	9	-0.33826	1.11959	1.04437	-0.22219	0.49936	-0.24676	0.65158	0.06954	-0.73673	-0.36685	1.01761	0.83639	1.00684	-0.44352	0.15022	0.73945	-0.54098	0.47668	0.45177	0.20371	-0.24691	-0.63375	-0.12079	-0.38505	-0.06973	0.0942	0.24622	0.08308	3.68	0
12	10	1.44904	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.04846	-1.72041	1.62666	1.19964	-0.67144	-0.51395	-0.09505	0.23093	0.03197	0.25341	0.85434	-0.22137	-0.38723	-0.0093	0.31389	0.02774	0.50051	0.25137	-0.12948	0.04285	0.01625	7.8	0
13	10	0.38498	0.61611	-0.8743	-0.09402	2.92458	3.31703	0.47045	0.53825	-0.55889	0.30976	-0.25912	-0.32614	-0.09005	0.36283	0.9289	-0.12949	-0.80998	0.35999	0.70766	0.12599	0.04992	0.23842	0.00913	0.99671	-0.76731	-0.49221	0.04247	-0.05434	9.99	0
14	10	1.25	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.32373	0.22767	-0.24268	1.20542	-0.31763	0.72567	-0.81561	0.87394	-0.84779	-0.68319	-0.10276	-0.23181	-0.48329	0.08467	0.39283	0.16113	-0.35499	0.02642	0.04242	121.5	0

Figure 3 – Screenshot of the Dataset's CSV file

The dataset is convenient to use because it has already been transformed using Principal Component Analysis (PCA). PCA is a dimensionality reduction algorithm that creates important variables/features through a combination of original variables, allowing us to better comprehend the dataset. The present dataset already includes features from V1 to V28, as well as the time and amount of the transaction and a class column to indicate whether or not the transaction is fraudulent (1 if fraudulent and 0 if normal). Although we cannot specify what each feature of V1 to V28 means, the transformation will help when processing, training, and testing the dataset using ML algorithms since it retains the necessary variables for calculations.

4. Experimental Study

4.1 Aim of the Experiment

The goal of the experiment is to help find metrics the following metrics:

- Confusion matrix
- Precision and Recall Score
- Accuracy and F-1 Score

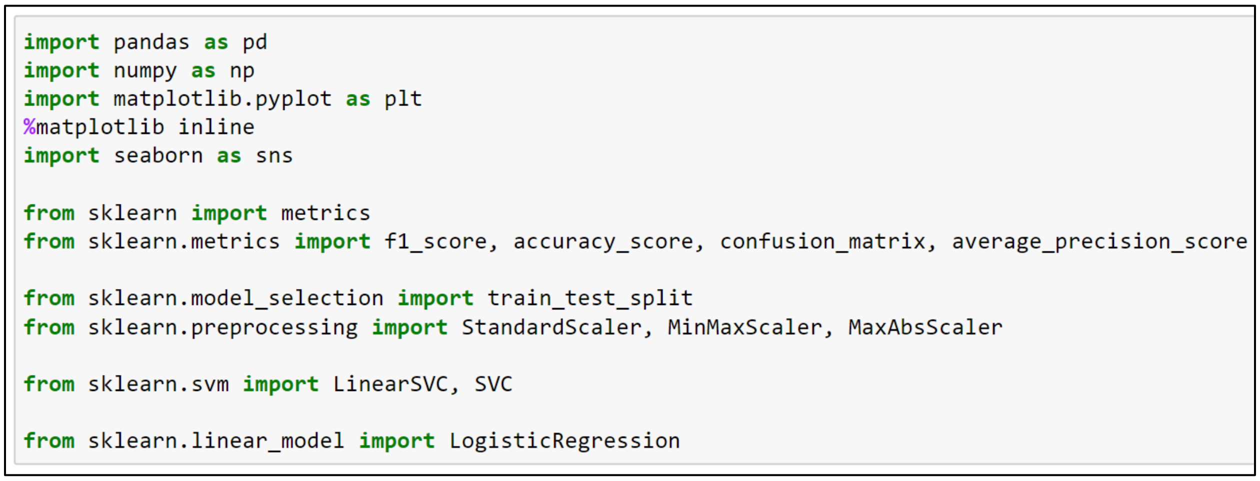
The aim of the experiment is to use the metrics mentioned to help decide which of the two machine learning algorithms (logistic regression or support vector machine) are faster and more efficient. Through comparing and evaluating these scores, the better algorithm for detecting fraud can be found.

4.2 Setup and Materials

For this experiment the setup included using the latest version of the programming language, Python 3. Using Python allowed the performance and development of the code to be much faster in getting the desired results. In python, multiple external libraries were used as this would help perform the experiment and get the metrics needed through ease.

4.3 Procedure

Firstly, before the dataset is dealt with, external libraries have to be imported.



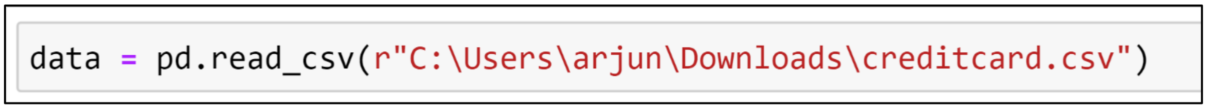
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix, average_precision_score

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

from sklearn.svm import LinearSVC, SVC

from sklearn.linear_model import LogisticRegression
```

Figure 4 – Importing all required libraries

4.3.1 Preparation and Verification of Dataset

The procedure for preparing and verifying the dataset before using it is as follows:

1. Open and access the dataset

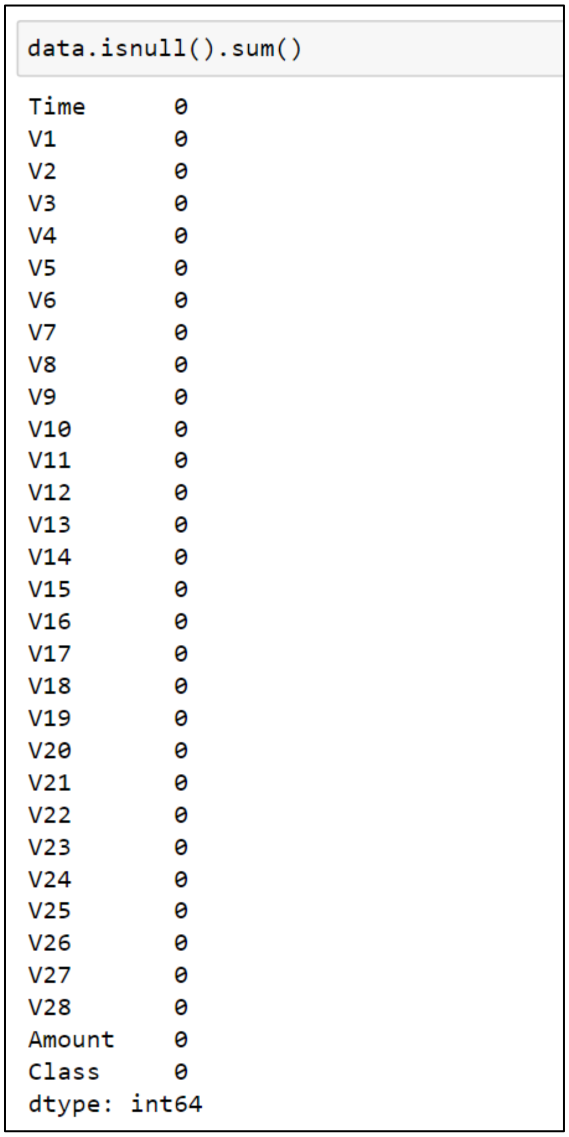


```
data = pd.read_csv(r"C:\Users\arjun\Downloads\creditcard.csv")
```

Figure 5 – Reading the CVS file containing the dataset

2. Changing the data type of the “Class” column from ‘int’ to ‘category’ as it shows if the transaction is in one of the two categories; normal or abnormal.

3. Ensuring no null values are present in the data frame. This is done so that a null value doesn’t interrupt the algorithm as a whole.



```
data.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

Figure 6 – Checking dataset to ensure no null values are present

4. Plotting a heatmap to show that there is no auto-correlation between two different variables.

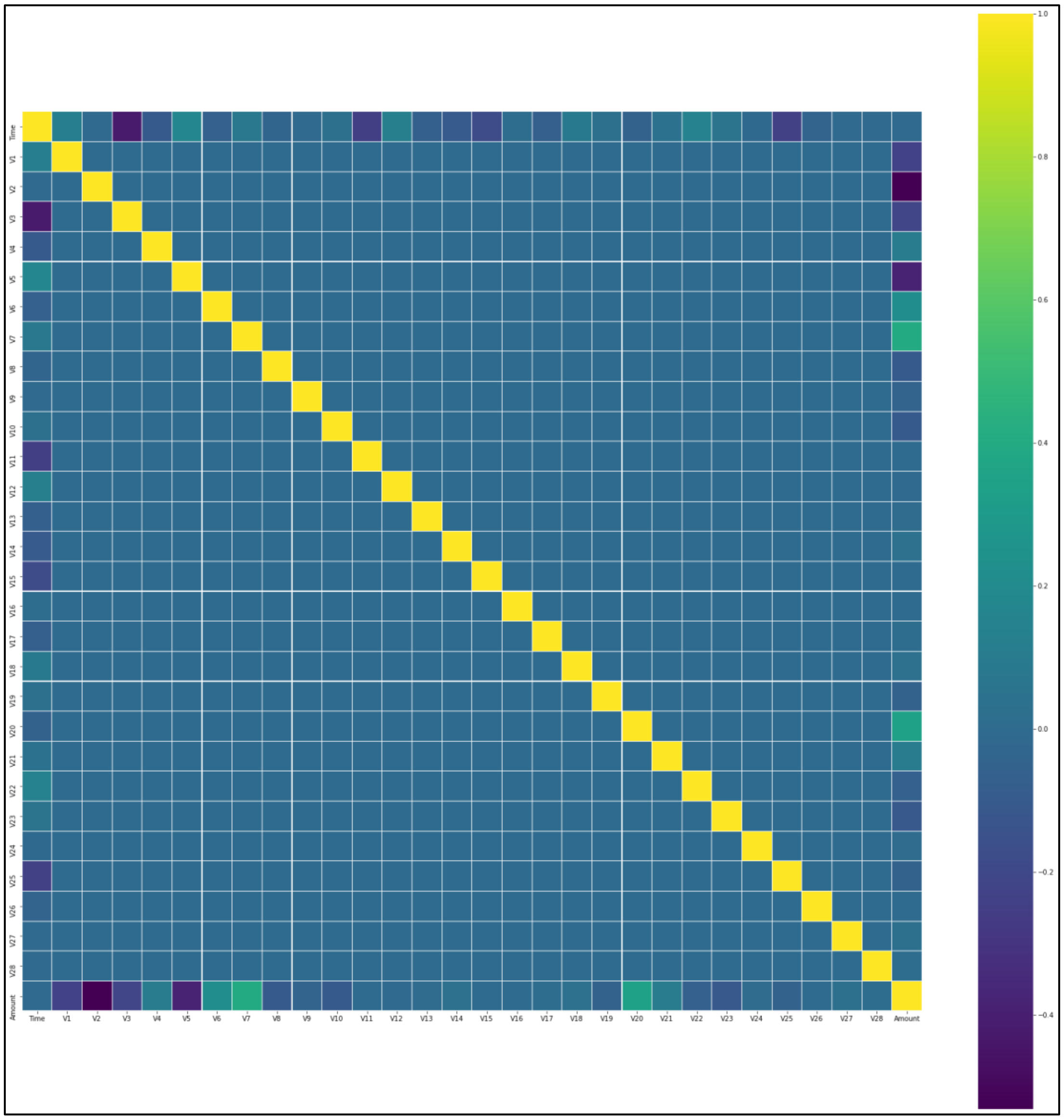
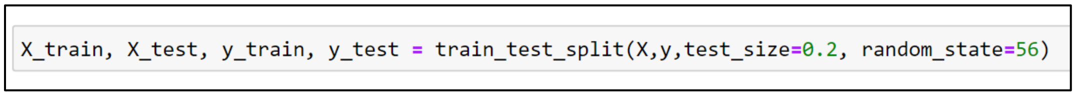


Figure 7 – Plotted Heatmap to check for autocorrelation

4.3.2 Splitting Dataset into Testing and Training

The steps for splitting the dataset into training and testing are as follows:

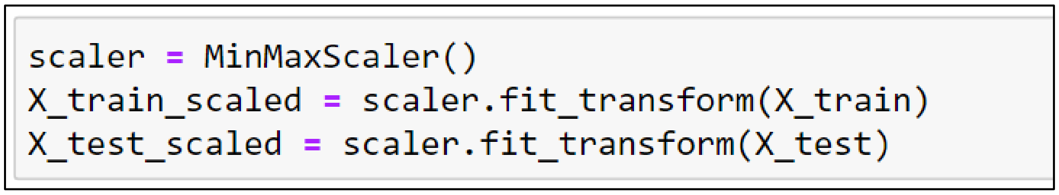
1. Separate all the x and y values. Y values consist of the "Class" column and X values consist of the independent variables.
2. Convert the X values into arrays as it will be easier to store many variables under one name.
3. Data is now ready and the X and Y values need to be split into train and test datasets.



```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=56)
```

Figure 8 – Scaling the X_train and X_test datasets

4. Once split the train and test datasets of X need to be transformed to avoid multiscale datasets so the datasets are all in single scale which makes calculations more accurate.



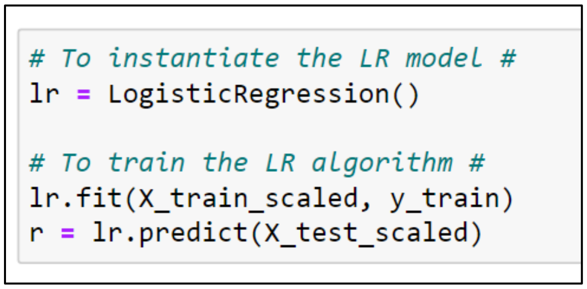
```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

Figure 9 – Scaling the X_train and X_test datasets

4.3.3 Training and Testing through Logistic Regression Algorithm

The steps to train and test the datasets through Logistic Regression are as follows:

1. Instantiate the logistic regression model to ready it for training the dataset.



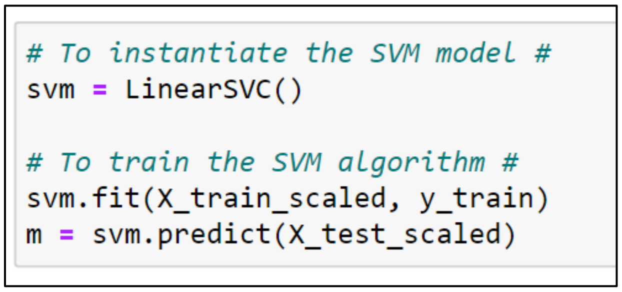
```
# To instantiate the LR model #
lr = LogisticRegression()

# To train the LR algorithm #
lr.fit(X_train_scaled, y_train)
r = lr.predict(X_test_scaled)
```

Figure 10 – code to instantiate and train the Logistic Regression algorithm

2. Train the algorithm using the training datasets.
3. Test the algorithm on the testing datasets and get the confusion matrix and predictive scores.

4.3.4 Training and Testing through Support Vector Machine Algorithm

1. Instantiate the support vector machine model to ready it for training the dataset.



```
# To instantiate the SVM model #
svm = LinearSVC()

# To train the SVM algorithm #
svm.fit(X_train_scaled, y_train)
m = svm.predict(X_test_scaled)
```

Figure 11 – code to instantiate and train the Support Vector Machine algorithm

2. Train the algorithm using the training datasets.
3. Test the algorithm on the testing datasets and get the confusion matrix and predictive scores.

4.4 Hypothesis

Now that we have a better understanding of the dataset, we must decide which of the two algorithms is more efficient. To determine this an experiment will be carried out to measure a variety of metrics; Accuracy score, F-1 score, Precision ratio, and Recall ratio. These scores will help in the comparison and determination of the more efficient algorithm out of logistic regression and support vector machine.

I hypothesize that:

- Support Vector Machine algorithm may be marginally better than the Logistic Regression algorithm in terms of **precision** due to support vector machine model’s ability to control overfitting better compared to logistic regression.
- Support vector machine’s **recall** score will significantly be higher than logistic regression as the logistic regression model is more reluctant to decide when elements in the dataset don’t show strong features
- The **accuracy** score of both the models will be very high as they both are powerful algorithms and accurately identify a very high number of elements of the majority class and that the minority classes results won’t be significant difference to make a change.
- The **F1 score** of the support vector machine model will be better due to it’s ability to use the dataset’s features in classifying the elements and future classification which makes the model better.

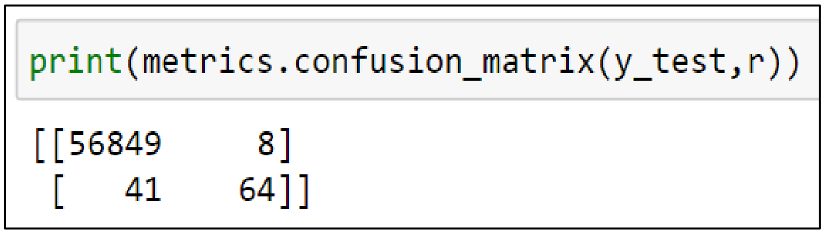
Overall, I believe that the support vector machine model will be the more efficient model out of the two when it comes to fraud detection.

4.5 Results

Upon performing the experiment and training the two ML algorithms, the results that will help decide which algorithm is the better and most efficient ones have been gathered and are displayed below.

4.5.1 Results of Logistic Regression

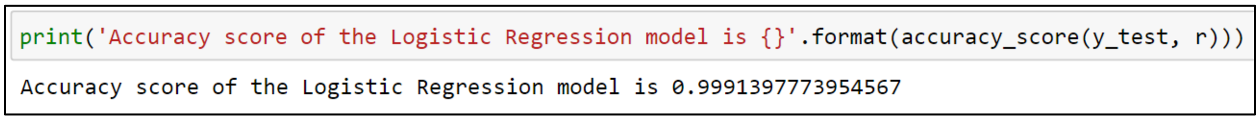
Confusion Matrix of Logistic Regression



```
print(metrics.confusion_matrix(y_test,r))

[[56849     8]
 [   41    64]]
```

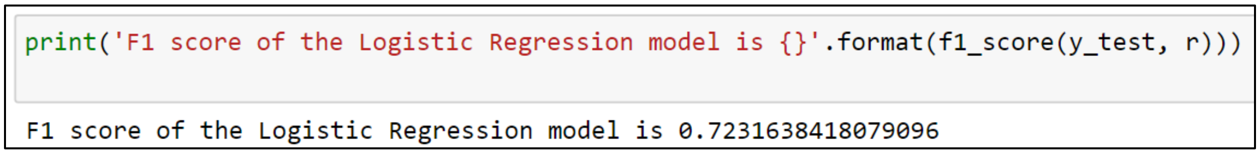
Figure 12 – confusion matrix of the classification done by the logistic regression algorithm

Accuracy and F1 Score of Logistic Regression



```
print('Accuracy score of the Logistic Regression model is {}'.format(accuracy_score(y_test, r)))

Accuracy score of the Logistic Regression model is 0.9991397773954567
```

Figure 13 – Accuracy score of the Logistic Regression Model



```
print('F1 score of the Logistic Regression model is {}'.format(f1_score(y_test, r)))

F1 score of the Logistic Regression model is 0.7231638418079096
```

Figure 14 – F1 score of the Logistic Regression Model

Precision and Recall Scores of Logistic Regression

Using the formula to calculate precision, the logistic regression algorithm's score is:

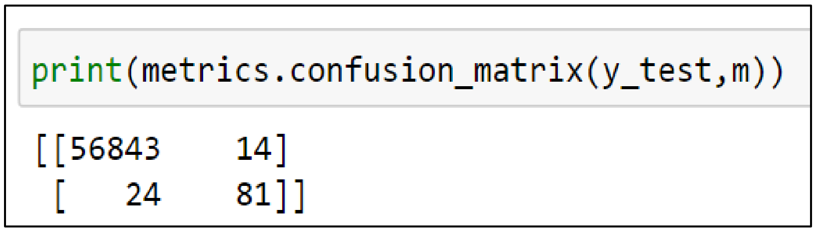
$$\frac{TN}{TN + \boldsymbol{FN}} = \frac{64}{64 + 8} = 0.88889 \approx 0.89\%$$

Using the formula to calculate recall, the logistic regression algorithm's score is:

$$\frac{TN}{TN + \boldsymbol{FP}} = \frac{64}{64 + 41} = 0.60952 \approx 0.61\%$$

4.5.2 Results of Support Vector Machine

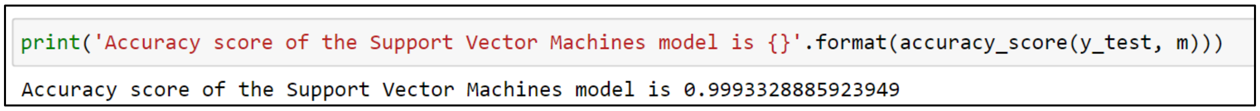
Confusion Matrix of Support Vector Machine



```
print(metrics.confusion_matrix(y_test,m))

[[56843    14]
 [   24    81]]
```

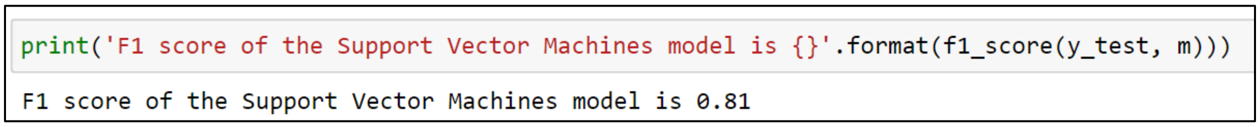
Figure 15 – confusion matrix of the classification done by the support vector machine algorithm

Accuracy and F1 Score of Support Vector Machine



```
print('Accuracy score of the Support Vector Machines model is {}'.format(accuracy_score(y_test, m)))

Accuracy score of the Support Vector Machines model is 0.9993328885923949
```

Figure 16 – Accuracy score of the Support Vector Machine Model



```
print('F1 score of the Support Vector Machines model is {}'.format(f1_score(y_test, m)))

F1 score of the Support Vector Machines model is 0.81
```

Figure 17 – F1 score of the Support Vector Machine Model

Precision and Recall Scores of Support Vector Machine

Using the formula to calculate precision, the support vector machine algorithm's score is:

$$\frac{TN}{TN + \boldsymbol{FN}} = \frac{81}{81 + 14} = 0.85263 \approx 0.85\%$$

Using the formula to calculate recall, the support vector machine algorithm's score is:

$$\frac{TN}{TN + \boldsymbol{FP}} = \frac{81}{81 + 24} = 0.77142 \approx 0.77\%$$

5. Discussion and Evaluation of Results

Score	Logistic Regression	Support Vector Machine
Precision Score	0.89	0.85
Recall Score	0.61	0.77
Accuracy Score	0.99	0.99
F1 Score	0.72	0.81

Table 1 – Concise List of Results

5.1 Discussion

My hypothesis of the recall score has been shown to be correct from the results above, with support vector machine's score significantly higher than logistic regression's score by 0.16. However, my hypothesis on support vector machine's precision score being better than logistic regression turns out to be wrong as the logistic regression in fact was better by 0.04.

As predicted the accuracy scores of both the ML models are very high with both being above 0.99, which is a really good score for an algorithm to have. Lastly my hypothesis for the F1 score of the support vector machine model to be better than the logistic regression model's was correct, but by a great extent of 0.11.

Although most of my predictions were correct, I was wondering as to why my hypothesis on the precision score was wrong. Upon further research I found out that the risk of overfitting was less in support vector machine due to it being a more complex model that was non-linear compared to logistic regression which was more prone to overfitting due to it being a linear model. This meant that although both the algorithms were able to reduce overfitting, logistic regression was more prone to it, thus its precision was higher; as out of all the transactions that were termed fraudulent between logistic regression and support vector machine, logistic regression had a greater True Negative-False Negative ratio than support vector machine did.

5.2 Evaluation

Having found and discussed the results, we can come to a consensus after the metrics that have been compared between the two ML algorithms that the support vector machine algorithm is more efficient than the logistic regression algorithm when identifying credit card fraud. The support vector machine model had a few advantages from the beginning due to the type of dataset that was used. As previously mentioned, an imbalanced dataset was used to help replicate the scenario that algorithms would have to tackle when used in the real world where majority of the transactions will be non-fraudulent. This meant that an support vector machine algorithm would be much suited when actually detecting fraud in the real world with instructed elements with several features.

Some of the reasons as to why support vector machine was a more efficient algorithm is because support vector machine tries to find the “best” margin that separates the classes. This helps reduce the risk of error when classifying. Logistic regression on the other hand has several decision boundaries that are weighted which is not as complex and useful as support vector machine’s method for detecting elements that belong to minor class labels.

A disadvantage that logistic regression had was that it is very sensitive to outliers when compared to support vector machine. This is because the sigmoid function of logistic regression causes it to diverge a lot quicker than support vector machine, which meant that being sensitive to outliers caused drawbacks in logistic regression’s algorithm. Another disadvantage of logistic regression which made it inefficient is that logistic regression model produces probabilistic values while support vector machine produced binary values (1 and 0). This meant that the logistic regression algorithm wouldn’t produce absolute predictions but would assume that the estimate it made was enough for a final decision. This property doesn’t work well when used with datasets that don’t have any missing values that make use lose confidence in the dataset. In the case of fraud detection, absolute predictions are necessary for it to be accurate as it is a binary problem and accurate predictions are required.

6. Conclusion

To conclude, this experiment aimed to compare the two ML algorithms through a variety of performance metrics that shows the abilities of both the models and where they exceled in and their drawbacks. Not only did the results of the experiments prove one of the ML models to be better than the other in terms of efficiency of detecting credit card fraud, but it also helped to prove the hypothesis that was previously made before the investigation.

To answer the research question as to how does the efficiency of logistic regression compare to the efficiency of support vector machine in detecting credit card fraud, I believe that the accuracy of both the algorithms would be neck to neck due to how advance the algorithms are. But, the efficiency of the algorithms purely depends on the type of dataset and in the dataset that was used for this exploration, the support vector machine model outperformed logistic regression model in and was more efficient in detecting fraud.

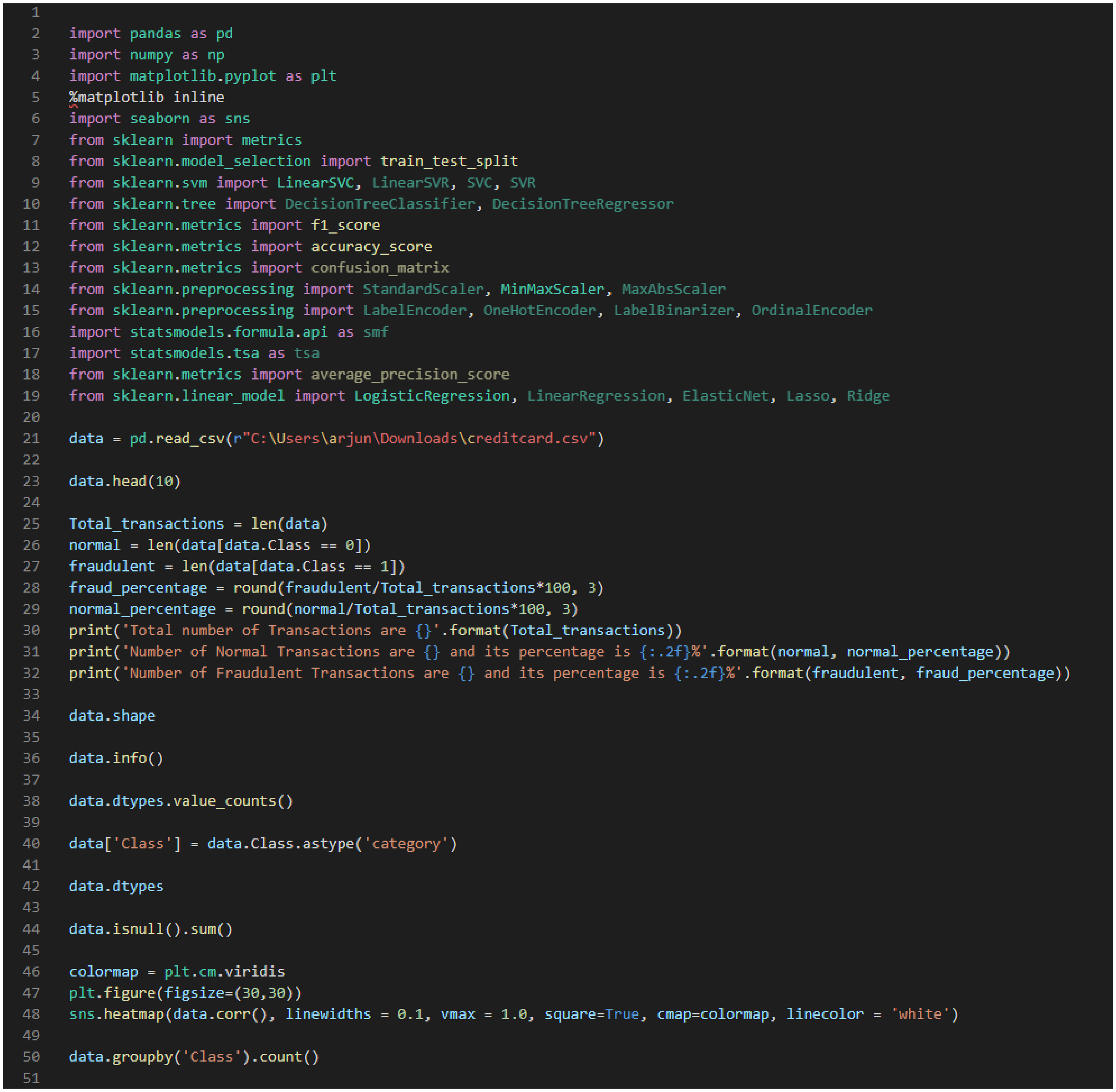
7. Bibliography

1. Carcillo, Fabrizio, Andrea Dal Pozzolo, Yann-Aël Le Borgne, Olivier Caelen, Yannis Mazzer, and Gianluca Bontempi. 2018. "SCARFF : A Scalable Framework for Streaming Credit Card Fraud Detection with Spark." *Information Fusion* 41 (May): 182–94. https://doi.org/10.1016/j.inffus.2017.09.005.
2. Carcillo, Fabrizio, Yann-Aël Le Borgne, Olivier Caelen, Yacine Kessaci, Frédéric Oblé, and Gianluca Bontempi. 2021. "Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection." *Information Sciences* 557 (May): 317–31. https://doi.org/10.1016/j.ins.2019.05.042.
3. "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy." 2018. *IEEE Transactions on Neural Networks and Learning Systems* 29 (8): 3784–97. https://doi.org/10.1109/TNNLS.2017.2736643.
4. Dal Pozzolo, Andrea, Olivier Caelen, Yann-Aël Le Borgne, Serge Waterschoot, and Gianluca Bontempi. 2014. "Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective." *Expert Systems with Applications* 41 (10): 4915–28. https://doi.org/10.1016/j.eswa.2014.02.026.
5. Lebichot, Bertrand, Yann-Aël Le Borgne, Liyun He-Guelton, Frédéric Oblé, and Gianluca Bontempi. 2020a. "Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection." In *Recent Advances in Big Data and Deep Learning*, edited by Luca Oneto, Nicolò Navarin, Alessandro Sperduti, and Davide Anguita, 1:78–88. Proceedings of the International Neural Networks Society. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-16841-4_8.

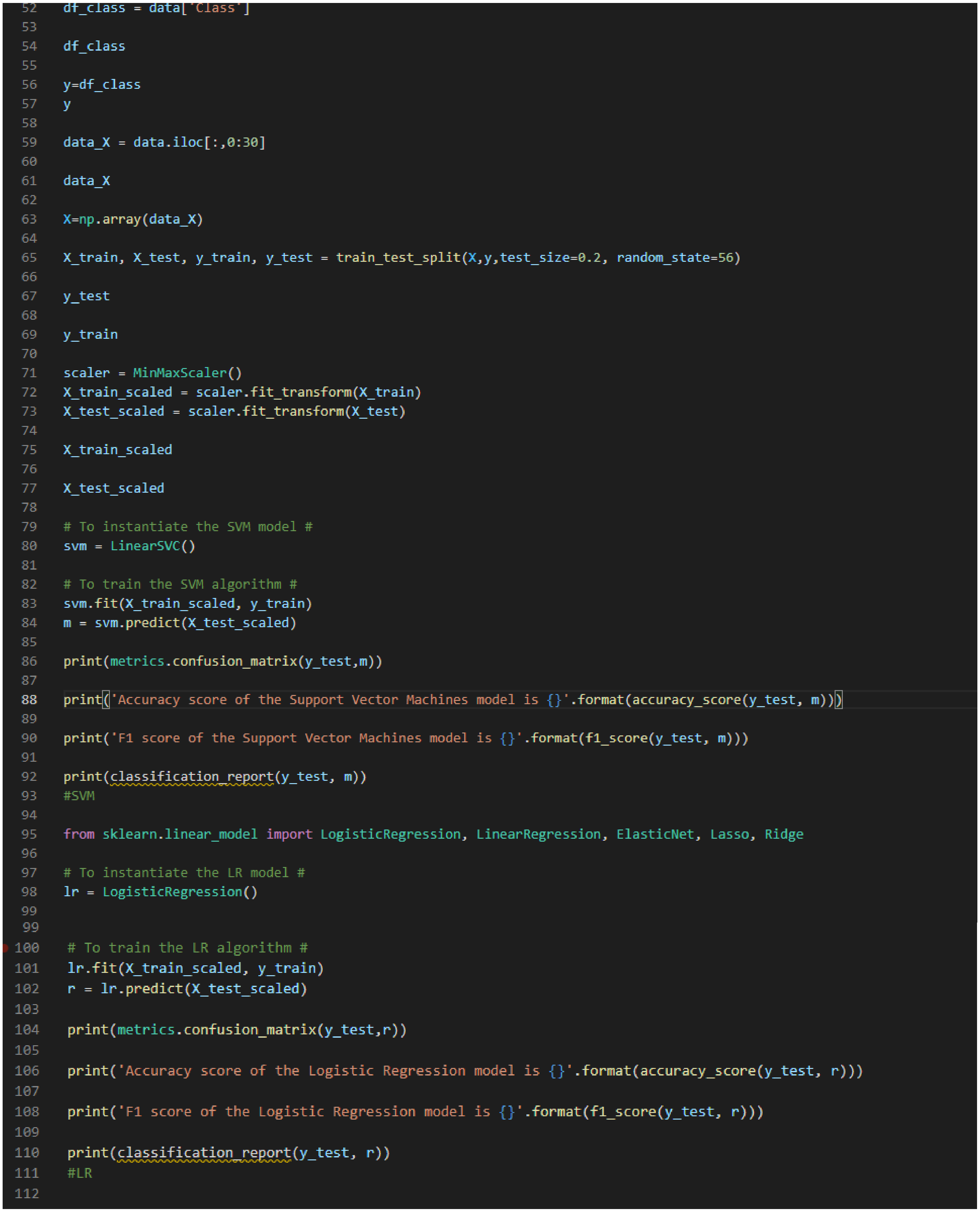
6. ———. 2020b. “Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection.” In *Recent Advances in Big Data and Deep Learning*, edited by Luca Oneto, Nicolò Navarin, Alessandro Sperduti, and Davide Anguita, 1:78–88. Proceedings of the International Neural Networks Society. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-16841-4_8.
7. Mitchell, Tom M. 1997. *Machine Learning*. McGraw-Hill Series in Computer Science. New York: McGraw-Hill.
8. Pozzolo, Andrea Dal, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. 2015. “Calibrating Probability with Undersampling for Unbalanced Classification.” In *2015 IEEE Symposium Series on Computational Intelligence*, 159–66. Cape Town, South Africa: IEEE. https://doi.org/10.1109/SSCI.2015.33.
9. Varmedja, Dejan, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, and Andras Anderla. 2019. “Credit Card Fraud Detection - Machine Learning Methods.” In *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 1–5. East Sarajevo, Bosnia and Herzegovina: IEEE. https://doi.org/10.1109/INFOTEH.2019.8717766.
10. Cucchiara, Andrew. (2012). Applied Logistic Regression. Technometrics. 34. 358-359. 10.1080/00401706.1992.10485291.
11. Le Borgne, Yann-Aël & Bontempi, Gianluca. (2021). Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook.
12. Uddin, Shahadat, Arif Khan, Md Ekramul Hossain, and Mohammad Ali Moni. 2019. “Comparing Different Supervised Machine Learning Algorithms for Disease Prediction.” *BMC Medical Informatics and Decision Making* 19 (1): 281. https://doi.org/10.1186/s12911-019-1004-8.

8. Appendices

Appendix A – Code Implemented for Experiment



```python

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC, LinearSVR, SVC, SVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, LabelBinarizer, OrdinalEncoder
import statsmodels.formula.api as smf
import statsmodels.tsa as tsa
from sklearn.metrics import average_precision_score
from sklearn.linear_model import LogisticRegression, LinearRegression, ElasticNet, Lasso, Ridge

data = pd.read_csv(r"C:\Users\arjun\Downloads\creditcard.csv")

data.head(10)

Total_transactions = len(data)
normal = len(data[data.Class == 0])
fraudulent = len(data[data.Class == 1])
fraud_percentage = round(fraudulent/Total_transactions*100, 3)
normal_percentage = round(normal/Total_transactions*100, 3)
print('Total number of Transactions are {}'.format(Total_transactions))
print('Number of Normal Transactions are {} and its percentage is {:.2f}%'.format(normal, normal_percentage))
print('Number of Fraudulent Transactions are {} and its percentage is {:.2f}%'.format(fraudulent, fraud_percentage))

data.shape

data.info()

data.dtypes.value_counts()

data['Class'] = data.Class.astype('category')

data.dtypes

data.isnull().sum()

colormap = plt.cm.viridis
plt.figure(figsize=(30,30))
sns.heatmap(data.corr(), linewidths = 0.1, vmax = 1.0, square=True, cmap=colormap, linecolor = 'white')

data.groupby('Class').count()

```



```python
df_class = data['Class']

df_class

y=df_class
y

data_X = data.iloc[:,0:30]

data_X

X=np.array(data_X)

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=56)

y_test

y_train

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

X_train_scaled

X_test_scaled

# To instantiate the SVM model #
svm = LinearSVC()

# To train the SVM algorithm #
svm.fit(X_train_scaled, y_train)
m = svm.predict(X_test_scaled)

print(metrics.confusion_matrix(y_test,m))

print('Accuracy score of the Support Vector Machines model is {}'.format(accuracy_score(y_test, m)))

print('F1 score of the Support Vector Machines model is {}'.format(f1_score(y_test, m)))

print(classification_report(y_test, m))
#SVM

from sklearn.linear_model import LogisticRegression, LinearRegression, ElasticNet, Lasso, Ridge

# To instantiate the LR model #
lr = LogisticRegression()


# To train the LR algorithm #
lr.fit(X_train_scaled, y_train)
r = lr.predict(X_test_scaled)

print(metrics.confusion_matrix(y_test,r))

print('Accuracy score of the Logistic Regression model is {}'.format(accuracy_score(y_test, r)))

print('F1 score of the Logistic Regression model is {}'.format(f1_score(y_test, r)))

print(classification_report(y_test, r))
#LR

```