Identifying Credit Card Frauds Using Machine Learning

Shrihan Tummala

Adlai E. Stevenson High School | Grade 10

Experimental

Table of Contents

Acknowledgements

I would like to express my thanks to my family for their support in helping me pursue this research. A special thanks to my sponsor, Mrs. Palffy, for investing her time to help me improve my project as well as support me.

Purpose

The purpose of this experiment is to determine the most accurate and effective algorithm for the binary classification of transactions into fraudulent and authorized. Credit card fraud has become more prevalent due to the rapid growth of technology usage. In spite of constant technological advances, credit card fraud continues to be a problem for many people. The costs of frauds such as these are staggering for companies, often amounting to billions of dollars. Many people have also been in a situation where they noticed a suspicious charge made to their account; this could have been a small amount or a large purchase, but in either case, the card may need to be canceled, a new card issued, and online accounts updated. As a result, it is critical to find an algorithm capable of accurately detecting credit card fraud before the problem escalates.

Hypothesis

If the dataset of credit card transactions (in PCA form for anonymization) is fit into a Decision tree, logistic regression, random forest, support vector machine, K-nearest neighbor, Naïve Bayes, or XGBoost classifier, then the algorithm with the highest percent accuracy will be the XGBoost classifier. This is because the XGBoost classifier makes the final prediction by constantly inserting new trees that predict residuals or errors of previous trees to then combine it with the previous trees to drastically increase the accuracy of the prediction (gradient boosting), which differs from other algorithms which construct their model with most or all of their data at once. Though this can lead to overfitting, the XGBoost classifier, with the use of the loss function and regularization parameters avoids overfitting as well as underfitting.

Experimental Components

Independent Variable: The machine learning algorithms that will be used to detect credit card frauds, specifically the logistic regression classifier, support vector machine classifier, decision tree classifier, Naïve Bayes classifier, and the XGBoost classifier.

Dependent Variable: The percent accuracy of the algorithm model's predictions on whether the credit card transaction is a fraud or not.

Constants: Variables that will be kept constant in the trial are the programming language used (Python 3.9.5), the computer used, the parameters, the software the program is run on (Jupyter Notebook), the libraries used (Pandas, NumPy, and SciKit-Learn, testing and training split ratio (80:20), and the dataset (Machine Learning Group - ULB Credit card frauds dataset).

Comparison Group: A K-Nearest Neighbor classification algorithm that makes a prediction on whether a transaction is legitimate or fraudulent. The KNN algorithm is also the industry standard for the classification of credit card transactions as it does not have much bias and does not make assumptions on the data points due to not being model-based.

Review of Literature

Machine learning has been popular in many prominent industries because of the ongoing developments of data-based software and computer processing over the previous decade. To this day, machine learning (ML) is necessary for services such as fraud detection. It has become vital to detect illegal and fraudulent activities since credit card fraud has resulted in considerable losses for people and corporations. The truth is that most fraudsters get away with their crimes. So improving algorithmic interactions with the data is required to enhance the capabilities of machine learning algorithms in many of these critical applications. It enables algorithms to replicate and expand on human observations, allowing for the quick resolution of several issues now linked with the digital era, such as the rise in fraudulent instances (*Machine Learning: What it is and Why it Matters*, n.d.).

Machine Learning

Machine learning is a subset of artificial intelligence that uses data to uncover patterns and unique structures in order to make decisions without the need for human involvement. These approaches are intended to allow machines to work autonomously without the need for precise programming, as the machines are simply given data from which they learn and adapt. (*Machine Learning: What it is and Why it Matters*, n.d.).

Supervised learning is a major subset of machine learning. These learning algorithms are trained using labeled data, in which the model is given the intended output as well as the input. The learning algorithm compares its output to the required inputs and outputs to identify defects in its model and improve it depending on the correct answer. Classification, regression, prediction, and gradient boosting are common supervised learning methods for using patterns to predict values. (*Machine Learning: What it is and Why it Matters*, n.d.).

Unsupervised learning is another well-known subset of Machine learning. In this case, the algorithm is not provided labeled input and is required to arrange it and develop conclusions on its own. The algorithm's primary goal is to discover patterns and structures in data. Some prominent approaches to applying unsupervised learning include k-means clustering, nearest-neighbor mapping, self-organizing maps, and singular value decomposition. (*Machine Learning: What it is and Why it Matters*, n.d.).

Semi-supervised algorithms are employed for the same purposes as supervised learning because of their utility in classification and regression issues; however, both labeled and unlabeled data are provided to it; yet, due to cost, more unlabeled data is often provided than labeled data. It makes use of labeled data to learn from unlabeled data. (*Machine Learning: What it is and Why it Matters*, n.d.).

Support Vector Machines

Algorithms for supervised classification classify and label data. Support Vector Machines are a popular approach for classifying data. The support vector machine (SVM) algorithm is a versatile binary classification algorithm. Beyond the standard X/Y prediction, this algorithm trains and classifies data within degrees of polarity. Support-vector machines focus on determining the optimal hyperplane or dividing border, between data belonging to distinct classes. The greatest distance between each data point is used to identify the optimum hyperplane. An additional dimension is introduced for more complicated data where the data cannot be classified linearly. Hyperplanes can have several dimensions and are defined using the formula:

$$\boldsymbol{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0}$$

Figure 1. SVM hyperplane equation (Source: Sirohi, 2019).

In Figure 1, β_0 represents the intercept, β_1 the first axis, β_2 the second axis, and so on, while *p* shows the number of dimensions, and *x* determines the position of a data point. Generally, on a cartesian plane, the two-pair coordinate system looks like (*x, y*), but in this case, it would be (x_1, x_2). The sum of the equation when substituted in with (x_1, x_2) equals zero, the point is on the hyperplane; if it is larger or less than zero, the point is that distance away from the hyperplane. When the data cannot be separated linearly, an extra transformation is used to transfer it onto an additional dimension. The algorithm's purpose is to linearly split the data with the hyperplane being in one dimension less (*p* - 1) (*Sirohi, 2019).*

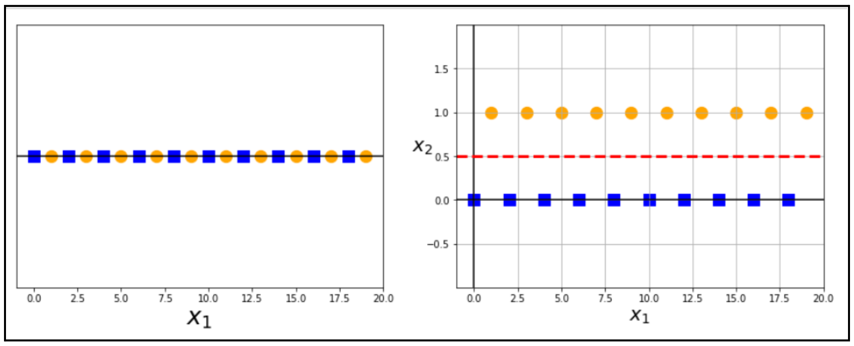


Figure 2. Graphs of SVM Transformation (Source: Wilimitis, 2018).

In Figure 2, a transformation maps a non-separable one-dimensional line onto a second dimension, which is then separated by a one-dimensional hyperplane (*p* - 1 dimensions). More sophisticated data can be mapped into three or more dimensions in the same way (Sirohi, 2019). However, when complicated transformations are necessary to transfer the data into many dimensions, the strategy becomes impracticable, resulting in unreasonable processing demands. The process can be bypassed by using kernels (Wilimitis, 2018).

The kernel function, while maintaining the original data points in a lower dimension, uses the dot product of vectors which is a measure of how closely two vectors align in terms of their pointing directions to map the data onto a high-dimensional feature space to avoid the use of complex transformations. The measure is a scalar number that can be used to compare the two vectors and determine the effect of moving one or both. The kernel function provides a quicker approach for support-vector machines to map data into a near-infinite number of dimensions by considerably lowering overall calculations (Wilimitis, 2018).

Even though the data can be separated linearly, there are an infinite number of possible hyperplanes because the hyperplane can always be positioned within a region or a range. In order to maximize the margin width (w) or the Euclidean distance from the nearest data points surrounding the separating region, support-vector machines choose a hyperplane that meets these requirements while still maintaining the basic structure of the data. As they affect the location of the hyperplane, these data points—known as support vectors—are prioritized over other data points. The next step is for the algorithm to establish the margin boundaries, which are a collection of lines offset in either direction from the primary hyperplane (Sirohi, 2019).

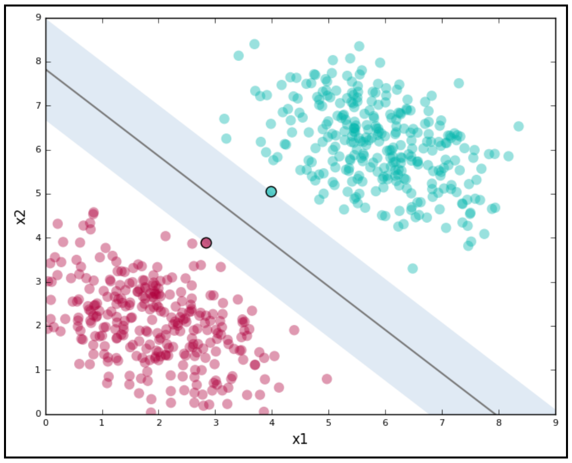


Figure 3. The SVM hyperplane (Source: Ghose, 2017).

The algorithm will seek to minimize the data in the margins, therefore the size of the margin may vary (Ghose, n.d.). Support-vector machines are vulnerable to overfitting because of how significantly the hyperplane and margins may change for a single outlier in the margin. With the slack parameter, support-vector machines can reduce the impact of such outliers. Support-vector machines can maximize the margin at the cost of mislabelling some data attributable to the slack variable (Sirohi, 2019). The algorithm can also define a penalty value for misclassifying a data point using the cost parameter, or C. With a lower C, the algorithm can misclassify a few data points to widen the margin since the penalty is lower, but with a higher C, the algorithm will select a hyperplane that classifies the data more accurately overall but with a smaller margin. (Ghose, n.d.). Whether the algorithm utilizes a soft margin hyperplane or a hard margin hyperplane depends on the values of the slack variable and cost combined. Some of the support vectors in a soft margin hyperplane may be misclassified or included in the margin to enhance the margin and confidence that the optimization has (Yildirim, 2020). In a hyperplane with a hard margin, optimization selects the hyperplane that best categorizes the data regardless of the margin (Sirohi, 2019).

Logistic Regression

Logistic regression is another popular supervised classification method algorithm. For binary classification, which divides data into only two unique groups (considered as class 0 or class 1), binomial logistic regression is utilized. Each data point is initially plotted on a Cartesian graph using the characteristics or vectors of the data as the X-axis and the related classification as the Y-axis; because the job is binomial, each data point's Y-value must be either 0 or 1. Utilizing linear regression, the line of greatest fit is obtained (Ponraj, 2020).

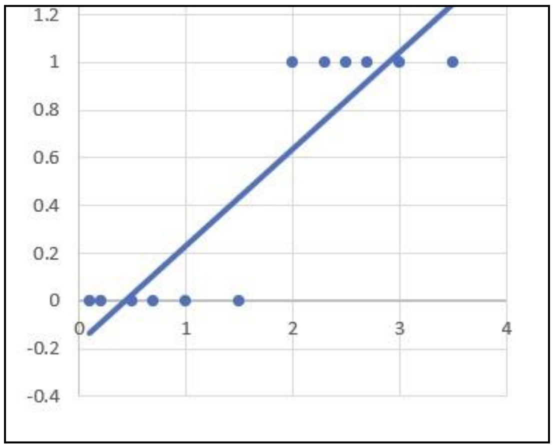


Figure 4. The linear function (Source: Ponraj, 2020).

When it comes to generating continuous results, the linear regression function is advantageous. Although the technique fails to give results that may be read as binary outcomes, particularly for values below 0 or above 1, when data is binary, as, in Figure 4, the outputs become unworkable and misrepresented. The sigmoid function, which converts any real integer to a decimal between 0 and 1, exclusive, is used to correct this. The likelihood of each data point belonging to one of the two groups is then shown on the Y-axis while the x-axis is kept constant (Ponraj, 2020).

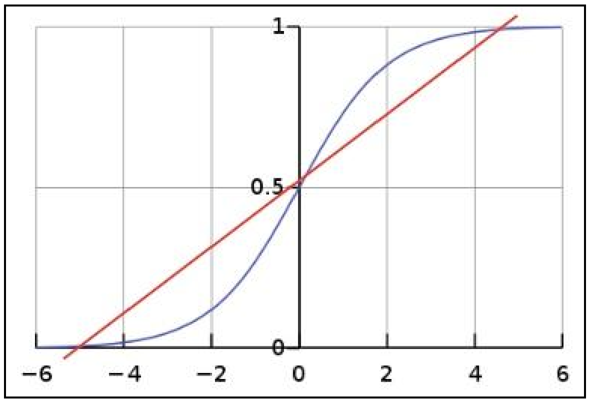


Figure 5. The sigmoid function (Source: Ponraj, 2020).

Figure 5 depicts the transformation of a linear function (red) into a sigmoid function (blue). The sigmoid function has an asymptotic range of 0 to 1. The algorithm can now readily categorize data since the Y-axis indicates probability; for any X-value, the matching Y-value on the function is the probability (P) that the data point belongs to class 1. If $P \geq 0.5$, the data point is classed as belonging to class 1, otherwise, if $P < 0.5$ it is classified as belonging to class 0 (Ponraj, 2020). Following are the mathematical functions of the linear function and sigmoid transformations:

$$z(x) = \theta_0 + \theta_1 x$$
$$p = 1 / (1 + e^{-z})$$

Figure 6. Linear and sigmoid equations (Source: Jordan, 2017).

Due to the reliance on the line of best fit, logistic regression algorithms are prone to outliers and overfitting (Jordan, 2017). Overfitting happens when a model is trained on an excessive quantity of data, resulting in the model accounting for the data's inherent unpredictability. The model could be more accurate and perform better on training data, but leads to less generalization, which leads to the model's decreased accuracy and susceptibleness to noise in real-world applications (Yildirim, 2020).

Decision Trees

Another popular supervised learning approach for categorization is the decision tree. It categorizes the data into numerous unique groups depending on its characteristics. Then it divides each group further, creating parts within the wider categories. It classifies data down to a certain group by repeating this procedure. For binary classification, all pathways would result in class 0 or class 1. (Wolff, 2020).

The algorithm creates divisions in order to ensure that the greatest number of samples are correctly categorized. Each aspect is considered and designated as a "branch" of the tree. The algorithm also prunes the tree, making it simpler by deleting portions that are connected to irrelevant attributes or that make superfluous divisions. Decision trees also take into account the data's inherent unpredictability, which is known as entropy (E), which is computed as:

$$E = \Sigma -p_i \log_2(p_i)$$

Figure 7. Entropy equation (Source: Roy, 2020).

Figure 7 shows that the summation's beginning constraint is i=1, and its limit is c, the number of classes. For binary classification, c would be 2. In the event that a random example was picked from the input data, each pi represents the chance that a class would be chosen. As a result, the entropy will be higher for a dataset where one class appears much more often than another, but a dataset where both categories have a more equal distribution would have a lower entropy (Roy, 2020).

Naive Bayes

Naive Bayes classifiers are a set of classification methods based on Bayes' Theorem that can be used for problems involving multiple classes, in this case, two, or even text classification. It calculates the possibility of whether a data point belongs within a certain category or not. In other words, assumes that the existence of one feature in a class is independent of the presence of other features in the data. The Bayes' Theorem calculates the likelihood of an event occurring given the chance of another event occurring (Kumar, 2022). The following equation represents Bayes' theorem mathematically:

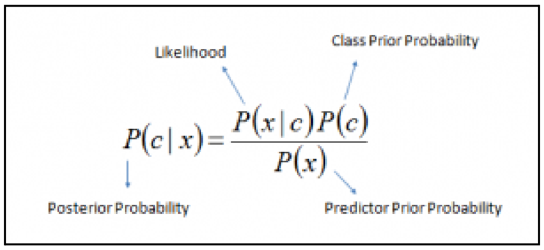

Likelihood
Class Prior Probability
$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

Figure 8. Bayes Theorem (Source: Ray, 2021).

In figure 8, *P(c|x)* denotes the posterior probability that class (*c*) is true given predictor (*x*), or the likelihood of class (*c*) given predictor (*x*). P(c), which is often given, stands for the prior probability of a class or the likelihood that the class is true. P(x) denotes the prior probability of the predictor or the likelihood of the predictor being true, and *P(x|c)* is the likelihood of the predictor given the class is true (Ray, 2021). In terms of variables, *x* and *y*, *X* would be the dependent feature vector (of size *n*) and *y* is a class variable (0 or 1), where *X* = ($x_1, x_2, x_3 \ldots x_n$) (Kumar, 2022). A similar approach is used by the Naive Bayes classifier to predict the likelihood of various classes based on various characteristics. The final prediction made by the model is the class with the highest posterior probability (Ray, 2021).

K-Nearest Neighbor

The supervised learning algorithm, K-nearest neighbors (KNN), is employed for both regression and classification problems. By computing the distance between the data used for testing and all of the training points, KNN tries to predict the appropriate class for the test data by checking the parameter, K, data points closest to the test data. The K value represents the number of closest neighbors. Distances between test points and training label points must be calculated. Given that it is computationally costly to update distance measures after each iteration, KNN is a slow learning method. To determine the most advantageous value of K, there are no pre-established statistical techniques. It is better to start by assigning a random K value as an initial value. Deciding on a low value of K results in unclear decision boundaries. The smoother the decision boundaries are, the better for classification the larger the K value is. The KNN method determines which classes of the "K" training data the test data will belong to, and the class with the highest probability is selected for prediction (Christopher, 2021).

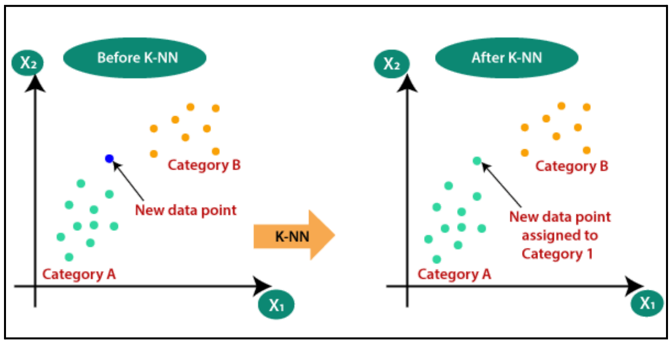
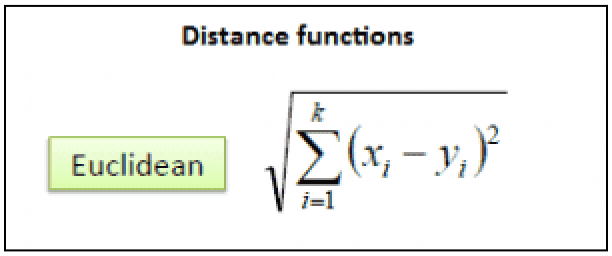


Figure 9. Graphs of KNN Transformation (Source: Christopher, 2021).

Figure 9 illustrates how the K-Nearest Neighbor algorithm operates. It determines the Euclidean distance between each of the K neighbors, then selects the K neighbors who are closest to them based on that distance. The number of data points from each category is tallied among these K neighbors, and the new data point is then allocated to the category with the greatest number of neighbors (Christopher, 2021). The Euclidean distance between data points is calculated with the formula:



$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Figure 10. Calculating Euclidean Distance (Source: Christopher, 2021).

In figure 10, the square root of the sum of the squared differences between a new point (x) and an existing point is used to determine the Euclidean distance (y) (Christopher, 2021).

<u>XGBoost</u>

XGBoost, also known as Extreme Gradient Boosting, is an ensemble learning method (Pawangfg, 2022) that combines predictions from numerous models to achieve higher performance. The three primary types of ensemble learning are bagging, stacking, and boosting. Bagging uses samples from the same dataset to fit several decision trees in order to calculate the average and provide a prediction. Through the process of stacking, the same data is used to create many models, which are then used to integrate their predictions into a single model (Brownlee, 2021). Boosting employs a sequence of weaker classifiers to add additional models and correct the flaws in the earlier models, resulting in a better final prediction (Pawangfg, 2022).

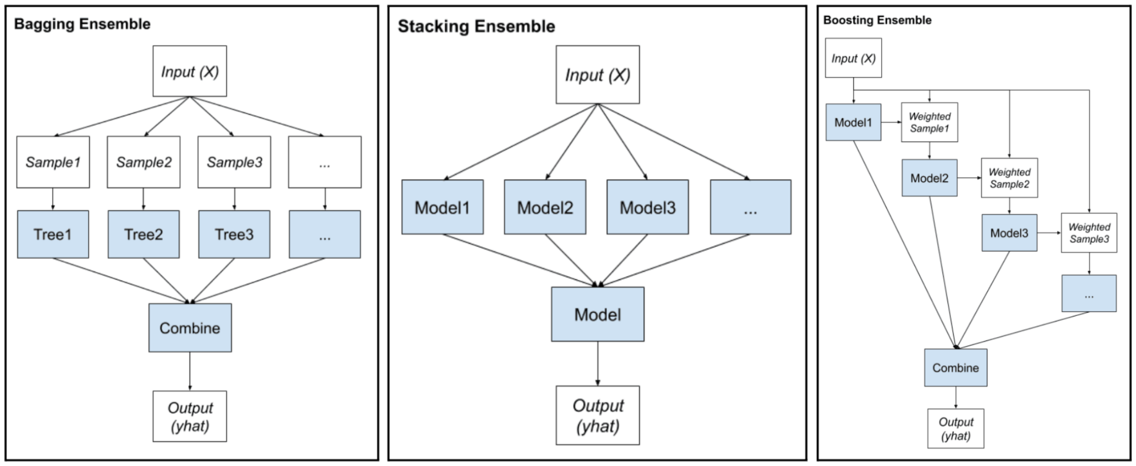


Figure 11. Different types of Ensemble Learning (Source: Brownlee, 2021).

Gradient boosting works by first repairing prior models' faults and then using the residual errors as labels to train a new prediction. XGBoost is a gradient-boosting implementation that generates a series of decision trees to which weights are allocated to all independent variables and then fed into the tree. The weight of variables, or the contribution of a class/variable to the loss function, predicted incorrectly by the tree is raised, and these variables are then fed to the second decision tree, and this process is repeated until the specified number of repetitions is reached (Pawangfg, 2022). The mathematical model would be in the form of:

$$\boxed{\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \epsilon \mathcal{F}}$$

Figure 12. Model of XGBoost (Source: Pawangfg, 2021).

In Figure 12, *K* represents the number of trees being used, *F* denotes the number of possible Classification and Regression Trees (CARTs), and *f* is the functional space of *F* or the set of functions between *F* (Pawangfg, 2021). The objective function of the above model is:

$$\boxed{obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)}$$

Figure 13. Objective Function of XGBoost (Source: Pawangfg, 2021).

Where the first term in Figure 13 is the loss function and the second is the regularization parameters (Pawangfg, 2021). The Loss Function evaluates how effectively the algorithm models the data; a greater loss function value indicates that the predictions are wrong, whereas a smaller loss function value indicates that the predictions are considerably more accurate (DataRobot, 2022). Regularization parameters are model calibration strategies that aim to reduce the loss function while avoiding overfitting or underfitting (Simplilearn, 2022). Instead of learning the full tree at once, the algorithm decreases loss first, then adds a new tree for improved optimization and efficiency (Pawangfg, 2021).

Principal Component Analysis (PCA)

Principal Component Analysis is a dimensionality reduction approach (Jaadi, 2021) that converts high-dimensional data into smaller dimensions while maintaining critical information (Pierre, 2021). PCA accomplishes this by transforming variables from bigger data sets into smaller ones that still retain information from the larger data set. Though this reduces accuracy because of data simplification, it allows machine learning algorithms to evaluate data substantially more effectively without having to worry about extraneous and irrelevant variables

(Jaadi, 2021). PCA initially standardizes the data range, preventing bias from forming between wide ranges of data and smaller ranges of data. This is accomplished through the utilization of the equation:

$$\boxed{z = \frac{value - mean}{standard\ deviation}}$$

Figure 14. Standardization Equation for PCA (Source: Jaadi, 2021).

Using the equation from Figure 14, all of the variables in a data collection are transformed into a more similar scale. The Covariance Matrix Computation is then used to determine whether or not there is a link between the data since correlated data might contain redundant information. The covariance matrix, which is a p * p symmetric matrix with p dimensions, is used to identify these relationships. It contains all the covariances associated with all possible pairings of the input variable. Whether the covariance is positive or negative determines if the variables are correlated or not. A positive sign implies that when one increases, the other increases as well, indicating that they are correlated, whereas negative means that when one increases, the other declines, indicating that they are inversely correlated (Jaadi, 2021). The formula for calculating the covariance is as follows:

For Population

$$\mathrm{Cov}(x,y) = \frac{\Sigma\,(x_i - \overline{x}) * (y_i - \overline{y})}{N}$$

For Sample

$$\mathrm{Cov}(x,y) = \frac{\Sigma\,(x_i - \overline{x}) * (y_i - \overline{y})}{(N - 1)}$$

Figure 15. Covariance Matrix Equation (Source: Jaadi, 2021).

An Eigenvector, a nonzero vector, changes when scaled by a scalar factor, eigenvalue, when that linear transformation is applied to it (The Nobles, 2020). The equation for finding the Eigenvector is:

$$Av - \lambda v = 0$$

Figure 16. Eigenvector Equation (Source: The Nobles, 2020).

The equation in Figure 16 shows the covariance matrix *A*, its eigenvector *v*, and the eigenvalue that satisfies *Av = λ*. The eigenvalues then are arranged in order to correlate to their associated eigenvector to start creating feature vectors. First, less significant components that have low eigenvalues are removed from the matrix, leaving just the remaining vectors, the feature vector. A feature vector is a matrix that contains the eigenvectors that remained after irrelevant vectors were removed. This aids in dimension reduction because if only *p* out of *n* (total) components are retained, the final data set will only have *p* dimensions. The data is remapped along the Principal Component Axes using the feature vector previously created by multiplying the transpose, or the interchanging of each row and column in the matrix, of both the original data and the feature vector (The Nobles, 2021).

$$\boxed{FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T}$$

Figure 17. The recasting of Data on PCA Axes (Source: The Nobles, 2020).

Credit Card Frauds

Credit card frauds are a form of identity theft that occurs when someone illegally obtains another person's information to make transactions or steal funds from the account. Application fraud and account takeover are the two main types of credit card fraud operations. Application fraud is the act of creating credit card accounts in someone else's name without their consent. This occurs when the offender obtains enough personal information to access the credit card freely or produce fake documents that seem legitimate. Account takeovers, on the other hand, involve the direct theft of an active account through having sufficient personal information of a person to alter the account, from which an offender can obtain a new credit card that can be used for future fraudulent use (*Legal Information Institute*, n.d.).

The Internet has developed dramatically in the last ten years. As a result, many payment services have expanded and become increasingly popular. As a result, cybercriminals targeting credit card transactions are far more prevalent. Credit card data encryption and tokenization are two measures in place to ensure the security of credit card transactions. While these strategies are usually successful, they do not provide comprehensive fraud protection for transactions with credit cards (Ileberi, Sun, & Wang, 2022). Machine learning models, which are among the most well-known ways of identifying fraudulent transactions, are alternative strategies that have also demonstrated efficacy in recognizing credit card fraud. Modern data-driven statistical and machine learning (ML) technologies can give statistics-like prediction models that output the likelihood of a transaction being fraudulent and overcome the aforementioned difficulty (Plakandaras, Gogas, Papadimitriou, & Tsamardinos, 2022).

Due to the world's constant desire to move practically every element of everyday life online, from buying to schooling to sustaining families, civilization is exposed to hackers and their assaults. While numbers like these may heighten consumer concerns about credit card theft, cardholders may find solace in the fact that the major credit card networks such as Visa, Mastercard, American Express, and Discover provide $0 liability protection. Basically, if someone is a victim of credit card fraud, they will not be made accountable for purchases they haven't made. As banks and credit card issuers bear a large portion of the financial risk associated with credit card fraud, they have a great interest in preventing fraud before it occurs. As a result, they are strengthening their security networks to tackle credit card theft front-on. To combat fraudsters attempting to make transactions in someone's name, the Visa credit card network employs Visa Advanced Authorization. When a transaction occurs, this anti-fraud detection system analyzes hundreds of pieces of data for risk using artificial intelligence (AI) and machine learning. Other companies aforementioned, employ technology that evaluates over 150 transaction factors using AI and machine learning to assist an issuer in making an informed choice to accept or refuse a transaction (Maxwell, 2022).

Materials

- ➢ 32-bit/64-bit PC (running Windows 8/10, macOS, or Linux).
- ➢ Access to the internet.
- ➢ Machine Learning Group - ULB's credit card frauds dataset (Retrieved November 26, 2022).
- ➢ Anaconda 3.8 individual edition with libraries such as Pandas, Numpy, and Scikit-Learn, and Jupyter Notebook.

Procedure

Install and Configure Anaconda 3.8 and Jupyter Notebook

1. Download the version of Anaconda 3.8 Individual Edition for your corresponding operating system, available from https://www.anaconda.com/products/individual.
2. Open the downloaded installer and follow the configuration prompts presented by the executable file. All necessary Python packages come with Anaconda Distribution.
3. Disable the PATH variable when installing Anaconda 3.8.

Run the program

4. Download the Credit card transaction dataset available from https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.
5. Extract the file from the downloaded zip folder.
6. Rename the downloaded file, "creditcard.csv."

7. Download the source code from https://github.com/VenkataST/Credit-Card-Detection-REACH.
8. On Windows, move the downloaded folder to the C drive under your user profile.
9. Move the dataset files (.csv files) to the same folder as the downloaded notebook, and ensure that the CSV files are not housed under a sub-folder.
10. Launch Jupyter Notebook using the shortcut created by Anaconda 3.8.
11. Navigate to the correct folder using the Jupyter Notebook interface.
12. Open the downloaded code (.ipynb file) via Jupyter Notebook.
13. Click the “Run All Cells” button available through the menu. Each algorithm receives a copy of the original dataset.
14. Record the output of the last 6 cells, which shows the algorithm used as well as the percent accuracy and confusion matrix, which is a 2x2 grid showing the number of true negatives, false positives, false negatives, and true positives when read left to right, top to bottom.
15. Repeat steps 13–14 for each trial. Previous algorithms are automatically deleted when the code is rerun.

Results

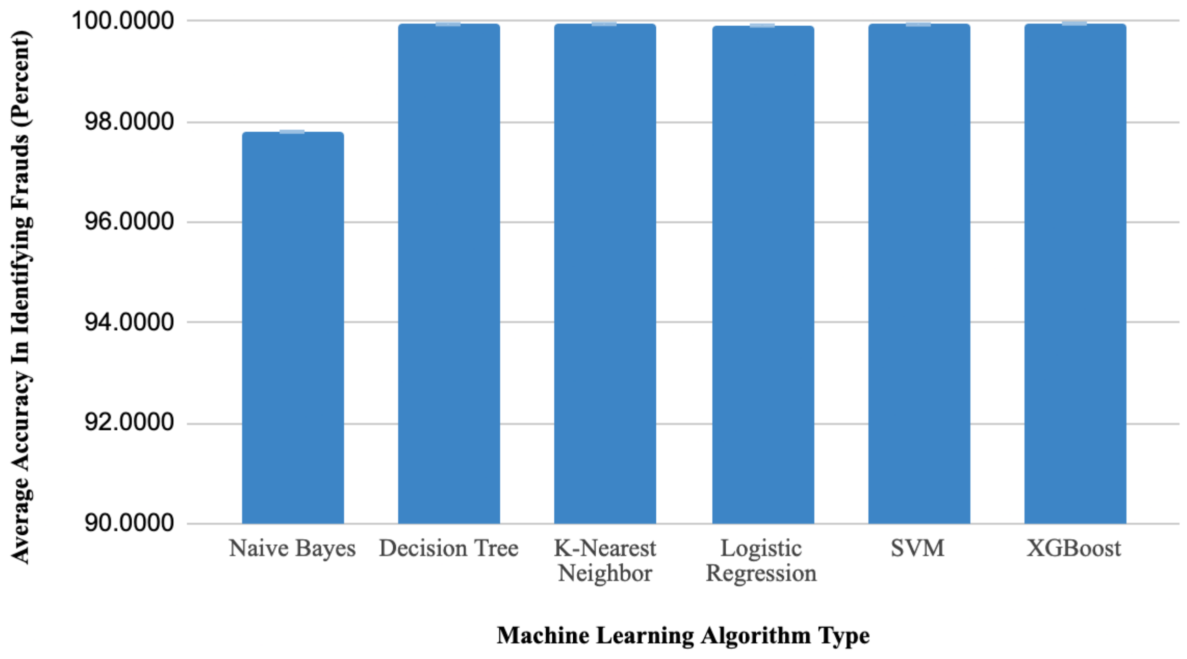
To evaluate the performance of each machine learning algorithm, k-fold cross-validation was conducted. For each machine learning algorithm, ten training and testing cycles using different data splits were carried out, after which it was erased. However, for one individual trial, all algorithms were evaluated on the same data splits.

Figure 18: Accuracy in Identifying Credit Card Frauds Versus Machine Learning Algorithm Type

Accuracy in Identifying Frauds (Percent)	Machine Learning Algorithm Type					
Trial	**Naive Bayes**	**Decision Tree**	**K-Nearest Neighbor**	**Logistic Regression**	**Support Vector Machines**	**XGBoost**
1	97.7345	99.9510	99.9528	99.9274	99.9383	99.9674
2	97.7708	99.9492	99.9474	99.9184	99.9420	99.9565
3	97.9722	99.9528	99.9510	99.9220	99.9474	99.9637
4	97.8289	99.9347	99.9474	99.9184	99.9383	99.9474
5	97.7563	99.9565	99.9547	99.9347	99.9492	99.9601
6	97.7962	99.9492	99.9492	99.9129	99.9401	99.9565
7	97.7291	99.9365	99.9492	99.9238	99.9365	99.9583
8	97.8597	99.9474	99.9438	99.9148	99.9383	99.9510
9	97.7981	99.9456	99.9456	99.9202	99.9436	99.9600
10	97.7817	99.9311	99.9401	99.8948	99.9293	99.9474
Average	97.8028	99.9454	99.9481	99.9187	99.9403	99.9568
Std. Dev.	0.0718	0.0084	0.0043	0.0105	0.0057	0.0066
Std. Err.	0.0227	0.0027	0.0014	0.0033	0.0018	0.0021

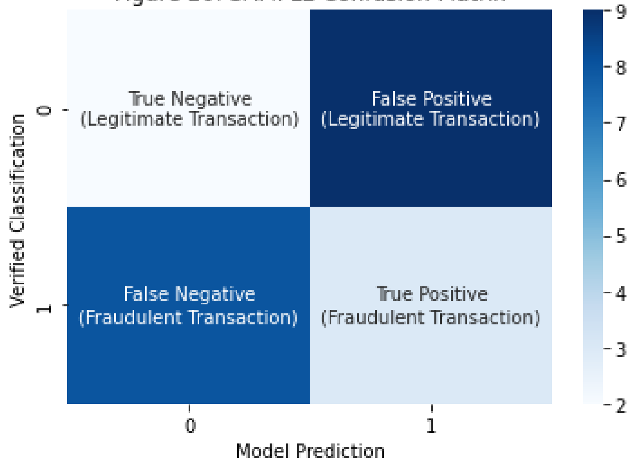
The performance of the models was largely comparable, except for the logistic regression classifier, which did not fare as well as the others. The XGBoost model exhibited the highest level of accuracy, approximately 99.96%, while the Naive Bayes model demonstrated the lowest level of accuracy at roughly 97.80%. The models demonstrated consistent performance, with an average standard deviation of 0.018%. Accuracy was ascertained by calculating the number of correctly identified data points in relation to the total number of data points used during testing.

Figure 19: Average Accuracy in Identifying Frauds Versus Machine Learning Type



The variability in performance between the models was minimal, with the average standard error measuring at a narrow 0.005%. The Naive Bayes classifier exhibited the most significant divergence in performance, making it both the least effective and least consistent. In terms of consistency, K-Nearest Neighbor performed the best among all the algorithms, with a 95% confidence interval ranging from 99.9469% to 99.9493%.

Figure 20: SAMPLE Confusion Matrix



The confusion matrix displayed shows the correct classifications (0 for true and 1 for fake) on the y-axis and the model's predictions on the x-axis. The correct predictions are represented in the true quadrants, while incorrect predictions are represented in the false quadrants. This information should be taken into consideration when evaluating figures 21-26.

Figure 21: Confusion Matrix for Naive Bayes

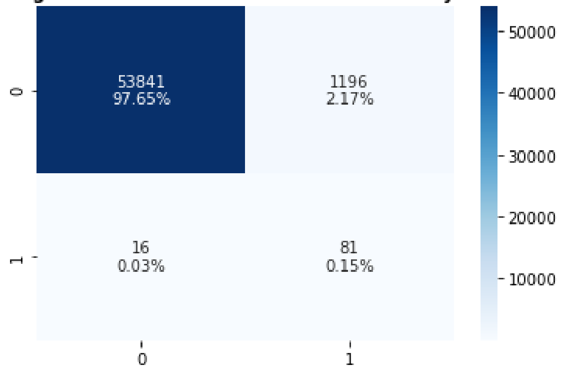


Figure 22: Confusion Matrix for Decision Trees

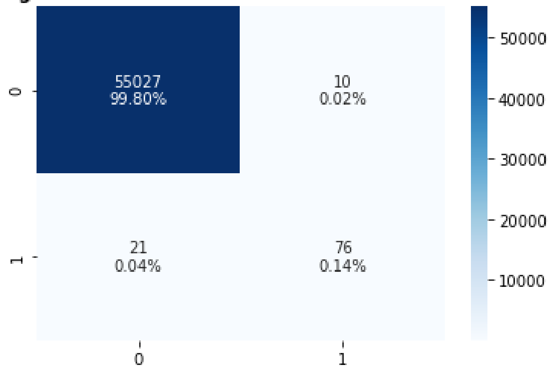


Figure 23: Confusion Matrix for K-Nearest Neighbor

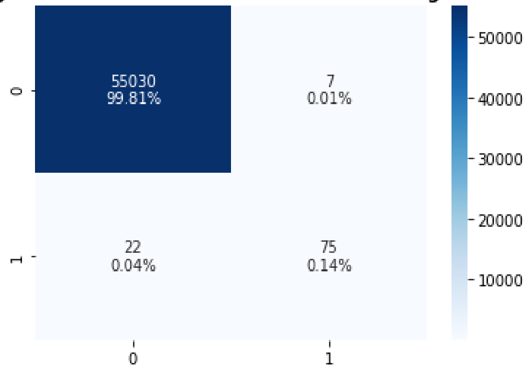


Figure 24: Confusion Matrix for Logistic Regression

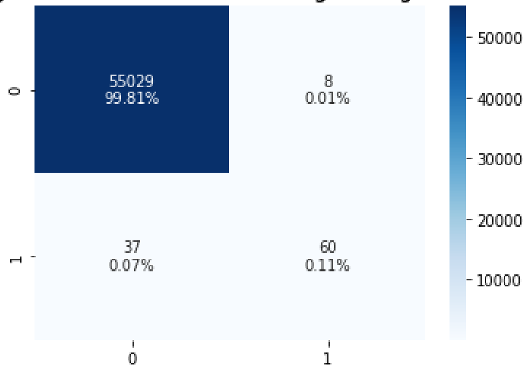


Figure 25: Confusion Matrix for Support Vector Machines

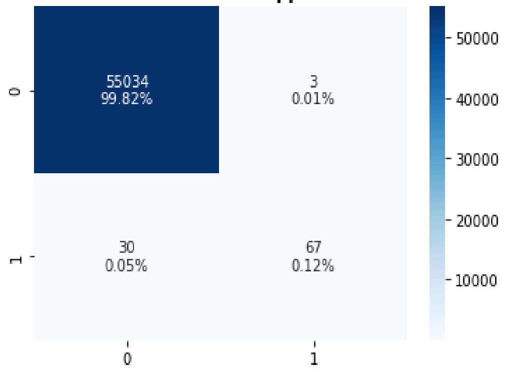


Figure 26: Confusion Matrix for XGBoost

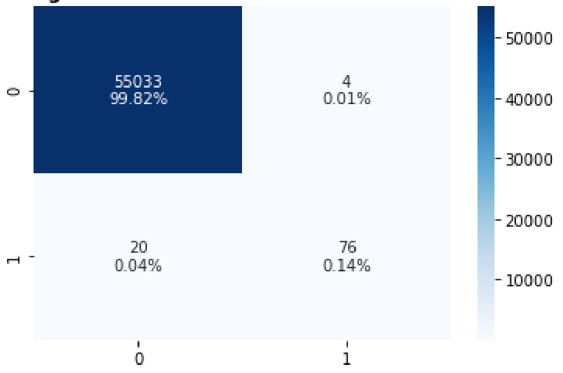
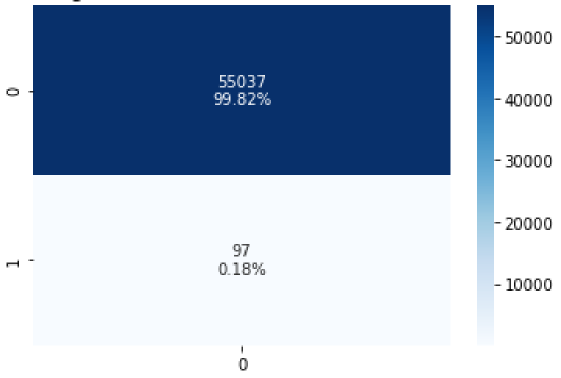


Figure 27: Total Distribution of Data



The confusion matrices illustrate the overall results of all ten trials, displaying the total amount of data classified in each quadrant as well as the proportion of data in each quadrant relative to the total. It appears that all models inaccurately identified more fraudulent transactions as legitimate than vice versa, except for the Naive Bayes, resulting in a higher number of false negatives compared to false positives. This may be due to the minuscule amount of data available for training and validation on fraudulent transactions, as shown in Figure 27, where the distribution is highly unbalanced as there is a significantly larger amount of legitimate data than fraudulent.

Discussion

A statistical analysis known as one-way analysis of variance (ANOVA) was conducted to compare the means of several groups. As the predetermined level of significance was set at 0.05, it can be concluded that at least one of the compared groups displayed a statistically significant difference from the others (p < 0.05). To further examine the differences between individual pairs of groups, two-tailed t-tests were conducted. The results of these tests are presented below.

Figure 28: Two-Tailed Unpaired T-Tests for Significance

Algorithm 2	**Algorithm 1**					
	Naive Bayes	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM	XGboost
Naive Bayes		t = 93.7270 p < 0.0001 SIGNIFICANT	t = 94.3162 p < 0.0001 SIGNIFICANT	t = 92.2095 p < 0.0001 SIGNIFICANT	t = 93.8464 p < 0.0001 SIGNIFICANT	t = 94.4700 p < 0.0001 SIGNIFICANT
Decision Tree			t = 0.9048 p = 0.3756 INSIGNIFICANT	t = 6.2791 p < 0.0001 SIGNIFICANT	t = 1.5887 p = 0.1303 INSIGNIFICANT	t = 3.3746 p < 0.05 SIGNIFICANT
K-Nearest Neighbor				t = 98.1939 p < 0.0001 SIGNIFICANT	t = 3.4546 p < 0.05 SIGNIFICANT	t = 3.4926 p < 0.05 SIGNIFICANT
Logistic Regression					t = 5.7172 p < 0.0001 SIGNIFICANT	t = 9.7148 p < 0.0001 SIGNIFICANT
SVM						t = 5.9832 p < 0.0001 SIGNIFICANT
XGBoost						

The t-tests conducted compared the performance of pairs of algorithms and evaluated whether the differences between the two were statistically significant. The results indicated that the majority of the compared algorithms displayed significantly different performance, as indicated by p-values less than 0.0001, which falls below the established significance threshold of 0.05.

The support vector machine and decision tree classifiers, as well as the decision tree and K-nearest neighbor classifiers, showed no significant difference in accuracy. However, these algorithms did not perform as well as the XGBoost classifier, which had the highest average accuracy and a statistically significant difference over all the other algorithms in identifying credit card fraud. Therefore, the other algorithms are not as optimal for this task due to their statistically different performances.

The performance of the various models excelled in terms of accuracy, with the Naive Bayes model exhibiting slightly lower accuracy compared to the others. All of the models tended to produce a greater number of false negatives, incorrectly classifying fraudulent cases as legitimate, rather than false positives. This led to lower recall scores, which are calculated as the number of true positives (TP) divided by the sum of the true positives and false negatives (TP + FN). A perfect recall score is 1, indicating that all positive cases (frauds) were correctly identified. Precision, on the other hand, is calculated as the number of true positives (TP) divided by the sum of the true positives and false positives (TP + FP). It reflects the proportion of cases correctly identified as positive out of all cases predicted as positive by the model.

Despite the notable accuracy of the various models, it is important to note that they were only evaluated on a single dataset due to availability constraints. While this dataset was substantial, comprising nearly 50,000 unique transactions, it had some limitations. For instance, the data was primarily sourced from Europe and predominantly dated to September 2013. Additionally, the class ratio was heavily imbalanced, with significantly more legitimate transactions compared to fraudulent ones. These factors may have slightly inflated the results compared to if the dataset had been more diverse and balanced. However, the use of k-fold cross-validation suggests that any such effects would have been minimal.

Conclusion

The purpose of this experiment is to identify the most efficient and accurate machine learning algorithm for detecting credit card fraud. With the increase in fraudulent activities and the detrimental financial impact it poses to financial institutions, identifying these illicit transactions has become an urgent priority for various platforms. In addition, the early detection of credit card fraud is also essential for safeguarding customers' personal information and protecting them from financial losses.

The initial hypothesis predicted that of the six tested algorithms, the XGBoost classifier would be the most successful in classifying data due to its ability to handle missing values and imbalanced datasets. This hypothesis was supported by the data from Figures 18 and 19, which showed that XGBoost had the highest accuracy among the tested algorithms for identifying credit card fraud. Other algorithms performed worse when accuracy was compared, and although the other algorithms may seem close in percentages, the t-tests from Figure 28 prove otherwise by showing that the difference in accuracy between XGBoost and the other algorithms is statistically significant. Overall, XGBoost is a highly effective and efficient algorithm for identifying credit card fraud.

One potential limitation of the experiment was the impurity of the folds used to evaluate the models' proficiency. A traditional k-fold cross-validation method involves pre-dividing the dataset and utilizing one partition for testing while utilizing the remaining partitions for training. However, this approach can decrease the number of trials or reduce the size of the testing sample per trial. As an alternative, this experiment randomly partitioned the dataset at the beginning of each trial. This methodology may have resulted in certain data points being utilized for testing multiple times, while other data points were not utilized at all. Despite this, the impact of this error appears to be negligible, as the standard deviations were low and each model was thoroughly evaluated. Furthermore, this suggests that data leakage did not occur.

There are various real-world applications for this project. Networks such as Visa and Mastercard have already begun using machine learning to detect credit card frauds, flagging suspicious transactions and prioritizing legitimate transactions. With the increase in online transactions and the rise of sophisticated fraud techniques, it is becoming increasingly important for companies to have robust systems in place for detecting and preventing credit card fraud. Credit card fraud is a major issue for financial institutions, costing them billions of dollars each year. Nearly 390,000 cases of credit card fraud were reported to the FTC (Federal Trade Commision) in 2021, and another 118,191 in the first quarter of 2022. Thus, using classifiers with a greater-than 99% chance of correctly identifying credit card frauds can greatly assist in mitigating the impact of such fraudulent activities.

This research has the potential for further expansion by incorporating data from a wider range of sources, as this research only utilized a dataset from Europe. Another avenue for expansion would be to test a combination of multiple algorithms to create a more sophisticated and blended model. For instance, a combination of the methods employed by K-Nearest Neighbor and XGBoost, which were among the best-performing algorithms, may lead to even greater effectiveness in detecting credit card fraud.

Reference List

Brownlee, J. (2021, April 26). A gentle introduction to ensemble learning algorithms. Machine Learning Mastery. Retrieved October 9, 2022, from https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/.

Christopher, A. (2021, February 2). K-Nearest Neighbor. Medium. Retrieved October 9, 2022, from https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4.

DataRobot. (2022, June 30). Introduction to loss functions. DataRobot AI Cloud. Retrieved October 9, 2022, from https://www.datarobot.com/blog/introduction-to-loss-functions/.

Ghose, A. (2017, August 24). Support Vector Machine (SVM) tutorial: Learning SVMs from examples. KDnuggets. Retrieved October 9, 2022, from https://www.kdnuggets.com/2017/08/support-vector -machines-learning-svms-examples.html.

Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for Feature Selection. *Journal of Big Data*, *9*(1),2–3. https://doi.org/10.1186/s40537-022-00573-8.

Jaadi, Z. (2021, April 1). A step-by-step explanation of principal component analysis (PCA). Built In. Retrieved October 9, 2022, from https://builtin.com/data-science/step-step-explanation -principal-component-analysis.

Jordan, J. (2017, October 18). Logistic regression. Jeremy Jordan. Retrieved October 9, 2022, from https://www.jeremyjordan.me/logistic-regression/.

Kumar, N. (2022, August 24). *Naive Bayes classifiers*. GeeksforGeeks. Retrieved October 10, 2022, from https://www.geeksforgeeks.org/naive-bayes-classifiers/.

Legal Information Institute. (n.d.). Credit Card Fraud. Legal Information Institute. Retrieved October 9, 2022, from https://www.law.cornell.edu/wex/credit_card_fraud#.

Machine learning: What it is and why it matters. Statistical Analysis System. (n.d.). Retrieved October 9, 2022, from https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-workings.

Maxwell, T. (2022, June 15). *How major credit card networks protect customers against fraud*. Bankrate. Retrieved October 9, 2022, from https://www.bankrate.com/finance/credit-cards/major-credit-card-networks-protect-against-fraud/.

Nobles, T. (2020, January 16). Understanding principle component analysis(PCA) step by step. Medium. Retrieved October 9, 2022, from https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9.

Pawangfg. (2022, July 11). XGBoost. GeeksforGeeks. Retrieved October 9, 2022, from https://www.geeksforgeeks.org/xgboost/.

Pierre, S. (2021, November 12). An introduction to dimensionality reduction in python. Built In. Retrieved October 9, 2022, from https://builtin.com/data-science/dimensionality-reduction-python.

Plakandaras, V., Gogas, P., Papadimitriou, T., & Tsamardinos, I. (2022). Credit card fraud detection with Automated Machine Learning Systems. Applied Artificial Intelligence, 36(1), 1–4. https://doi.org/10.1080/08839514.2022.2086354.

Ponraj, A. (2020, July 9). Logistic regression part I-transformation of linear to logistic. Medium. Retrieved October 9, 2022, from https://medium.com/analytics-vidhya/logistic-regression-part-i-transformation-of-linear-to-logistic-395cb539038b.

Ray, S. (2021, August 26). Learn naive Bayes algorithm: Naive Bayes classifier examples. Analytics Vidhya. Retrieved October 9, 2022, from https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

Roy, A. (2020, November 6). A dive into decision trees. Towards Data Science. Retrieved October 9, 2022, from https://towardsdatascience.com/a-dive-into-decision-trees-a128923c9298.

Simplilearn. (2022, September 9). Regularization in machine learning || Simplilearn. Simplilearn.com.

Retrieved October 9, 2022, from https://www.simplilearn.com/tutorials/machine-learning-tutorial/regularization-in-machine-learning.

Sirohi, K.. (2019, August 5). Support Vector Machine (detailed explanation). Towards Data Science. Retrieved October 9, 2022, from https://towardsdatascience.com/support-vector-machine-support-vector-classifier-maximal-margin-classifier-22648a38ad9c.

Wilimitis, D. (2018, December 12). The kernel trick. Towards Data Science. Retrieved October 9, 2022, from https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f.

Wolff, R. (2020, August 26). 5 types of classification algorithms in machine learning. MonkeyLearn Blog. Retrieved October 9, 2022, from https://monkeylearn.com/blog/classification-algorithms/.

Yildirim, S. (2020, June 1). Hyperparameter tuning for support vector machines– C and gamma parameters. Towards Data Science. Retrieved October 9, 2022, from https://towardsdatascience.com/Hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167.