Credit Card Fraud Detection and Analysis (INSE 6180)

Submitted to:
Dr. Arash Mohammadi

Submitted By:

Student Name	**Student ID**	**Email ID:**
Shivraj Patil	40234687	shivrajrameshpatil@gmail.com
Sai Venkata Lakshmi Soumya Challa	40231379	soumyasaic@gmail.com
Saeedeh Moghimi	40184202	moghimi.saeedeh@gmail.com

Table of Contents

Table of Figures

Acknowledgement

We would like to express our gratitude to Prof. Arash Mohammadi for providing us with this opportunity to do a project on this topic. We appreciate his excellent instruction and ongoing assistance with our project. We could never address many issues and misconceptions without his assistance. We would like to thank all the internet communities and websites from which we received advice and support.

1- Introduction

1-1 What is Credit Card Fraud Analysis?

The number of online payment options has expanded thanks to e-commerce and numerous other websites, raising the possibility of online fraud. As fraud rates rose, researchers began utilizing various machine-learning techniques to identify and evaluate scams in online transactions. The primary objective of the study is to create and implement a unique fraud detection algorithm for streaming transaction data to analyze historical customer transaction information and extract behavioral patterns. wherein cardholders are grouped according to the value of their transactions. Later, various classifiers are individually trained over the groups. The classifier with the highest rating score can then be selected as one of the most effective ways to detect fraud.

1-2 Mission

The aim of our project is to design and develop a model to detect weather the transaction is fraudulent or not based on the attributes in the data set.

2- Problem Statement

2-1 What is the problem?

Digital payments are evolving, but so are cyber criminals. More than 5 million data are stolen every day, according to the Data Breach Index, a worrying statistic that demonstrates fraud is still prevalent for both Card-Present and Card-not Present type of payments. Fraud detection is difficult in today's digital world, when trillions of Card transactions occur every day.

2-2 The role of Fraud Analysis in Security

To do this, we may train a machine learning model that can instantly spot any deviations from normal user behaviour and transactional patterns. ML algorithms can reduce the risk of fraud and enable more secure transactions by identifying anomalies, such as a sudden spike in transactional amount or geographical change.

3- Implementation Software

During the project, we used the programming language python in the editor tool google collab. We encounter many small problems in python programming. With the group efforts, we sufficiently solve the problem, and obtain the correlated results.

4- Methodology

4-1 Introduction

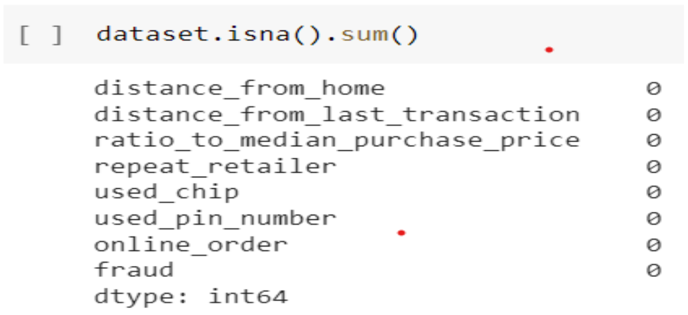
Credit Card Fraud Detection and Analysis will be performed based on the steps mentioned below:

- A large dataset in the form of a CSV file will be used in the project.
- The dataset consists of 1000000 rows and 8 columns.
- The dataset has 7 independent columns and 1 dependent column.
- The dataset is divided into 80% training data and 20% test data.
- After the model is trained, the model can be used to check whether a transaction is fraud or not based on certain attributes.
- We have also implemented a data poisoning attack on the Support Vector Machine Classifier which decreases the accuracy of the model.

4-2 How it operates?

4-2-1 Data Cleaning Phase

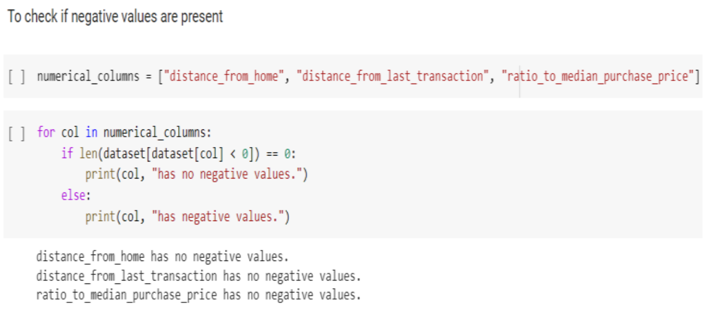
A null value in a relational database means when the value in a column is unknown or missing. We checked our dataset for null values as shown below in the snippet. We did not find any null values.



```
[ ]  dataset.isna().sum()

     distance_from_home                0
     distance_from_last_transaction    0
     ratio_to_median_purchase_price    0
     repeat_retailer                   0
     used_chip                         0
     used_pin_number                   0
     online_order                      0
     fraud                             0
     dtype: int64
```

We also checked for the negative values in the data set as the attributes distance from home, distance from the last transaction, and ratio to median purchase price parameter can't be negative and we did not find any.



To check if negative values are present

```
[ ] numerical_columns = ["distance_from_home", "distance_from_last_transaction", "ratio_to_median_purchase_price"]

[ ] for col in numerical_columns:
        if len(dataset[dataset[col] < 0]) == 0:
            print(col, "has no negative values.")
        else:
            print(col, "has negative values.")

    distance_from_home has no negative values.
    distance_from_last_transaction has no negative values.
    ratio_to_median_purchase_price has no negative values.
```

4-2-2 Exploratory Data Analysis:

The first step is to find out the percentage of fraud and non-fraud cases based on different attributes. We use Normal Distribution to find out the mean, median and mode of the curve. Normal Distribution is a probability distribution that is symmetric about the mean and sometimes referred to as the Gaussian distribution. It demonstrates that data that are close to the mean occur more frequently than data that are far from the mean. The standard deviation determines the width of the curve in a normal distribution, which depicts a symmetrical representation of data around its mean value. It is represented graphically as the "bell curve."

Skewness gauges a distribution's degree of symmetry. The skewness of the normal distribution is zero, and it is symmetrical. If the distribution of a data set has a skewness less than zero, or negative (left-skewness), then the left tail of the distribution is longer than the right, while positive skewness (right-skewness), denotes the opposite.

The mean, median and mode of the dataset are at the same point of the curve as the data is normally distributed for the attributes as shown below in the graph.

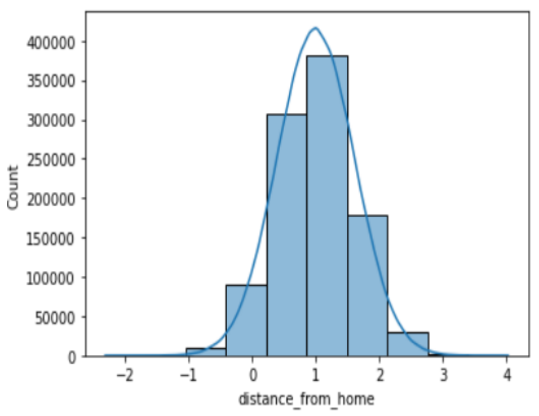


Fig-1: Normal distribution curve for attribute distance_from_home

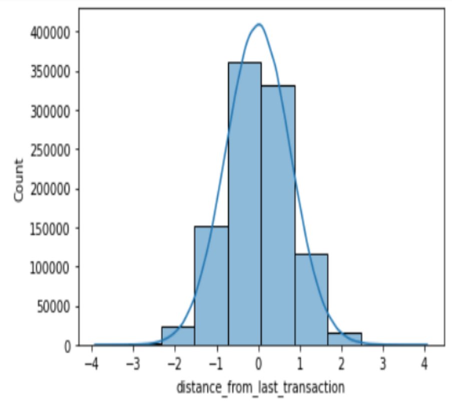


Fig 2: Normal distribution curve for attribute distance_from_last_transaction

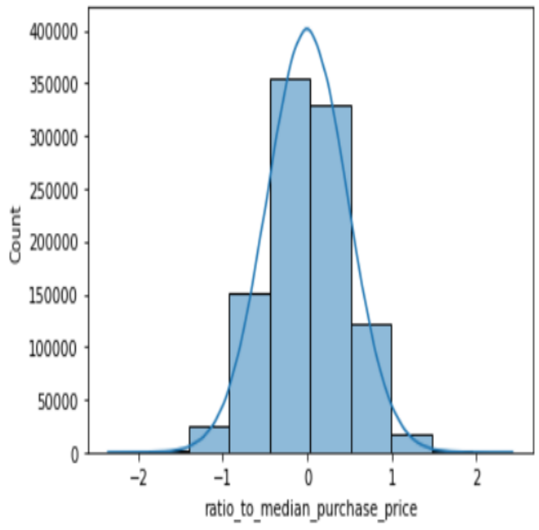
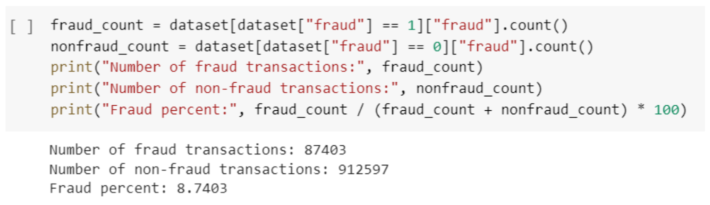


Fig 3: Normal distribution curve for attribute ratio_to_median_purchase_price.

We found the Number of fraud and non-fraud transactions with fraud percent using the code as shown below.


```
[ ]  fraud_count = dataset[dataset["fraud"] == 1]["fraud"].count()
     nonfraud_count = dataset[dataset["fraud"] == 0]["fraud"].count()
     print("Number of fraud transactions:", fraud_count)
     print("Number of non-fraud transactions:", nonfraud_count)
     print("Fraud percent:", fraud_count / (fraud_count + nonfraud_count) * 100)

     Number of fraud transactions: 87403
     Number of non-fraud transactions: 912597
     Fraud percent: 8.7403
```

We also use the pie chart to find the fraud and non-fraud percentages. The pie charts are one of the most popular types of data visualizations is the pie chart. Additionally, they are among the most widely criticized and misapplied. A pie chart illustrates a part-to-whole relationship in your data.

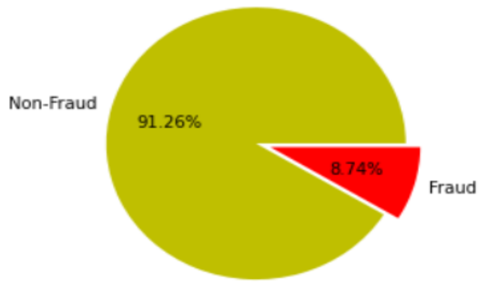


Fig 4: Percentage of fraud and non-fraud transactions

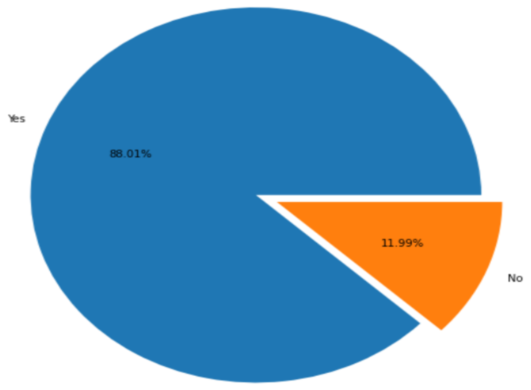


Fig 5: Percentage of was the retailer repeated or not.

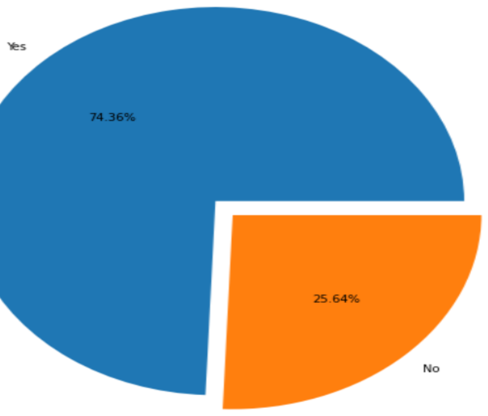


Fig 6: Percentage of was the credit card chip used

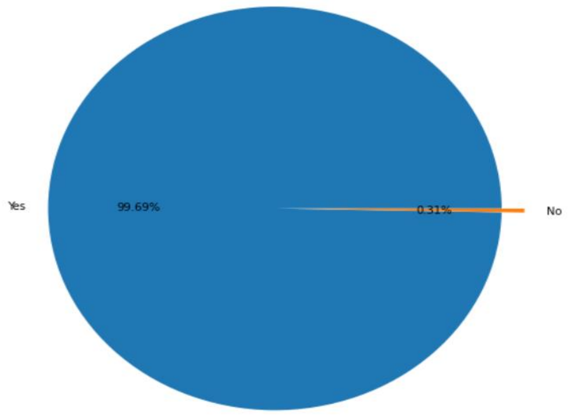


Fig 7: Percentage of credit card pin number used

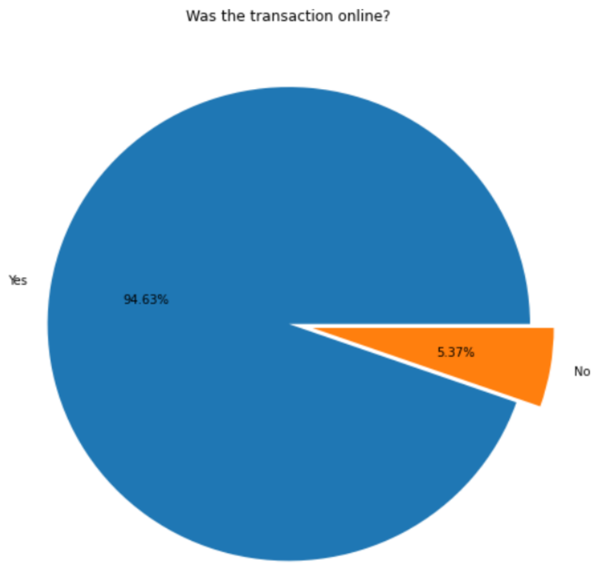
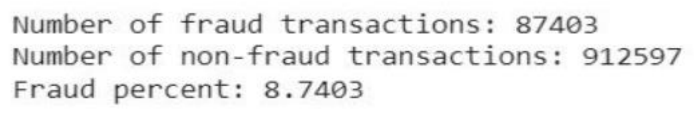


Fig 8: Percentage of was the transaction online

As we can see that there is an imbalanced data for fraud and non-fraud.



```
Number of fraud transactions: 87403
Number of non-fraud transactions: 912597
Fraud percent: 8.7403
```

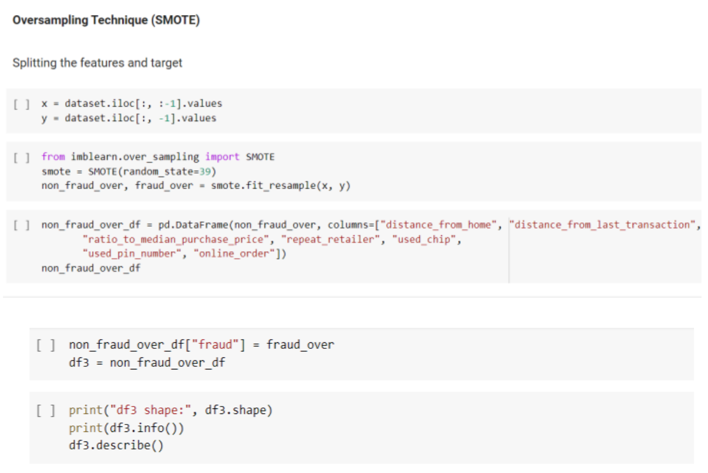
To balance this, we use the over sampling technique called SMOTE to balance the data before training the model.

4-2-3 Oversampling Technique:

Oversampling is the reverse of under sampling since, with oversampling, the class is balanced by simply adding additional points to the minority class rather than taking them away from the majority class. Now that we don't have any additional data, you might be wondering how we can add new points to the minority class. In order to respond to this issue, I would reply that we either produce points at random or using mathematical calculations for the minority class.
There are various oversampling methods, including Random oversampling and SMOTE, to mention a couple. Here, we use SMOTE Oversampling technique to balance the data before training the model.

Random Oversampling:
As the name implies, this method involves picking random data points from the minority class and duplicating them in order to enhance the minority class's number of data points. but is said to be the strongest of all. This approach is regarded as the most fundamental oversampling strategy. The variance of the dataset is artificially reduced because the points are chosen at random.

SMOTE Oversampling:
Synthetic Minority Oversampling Technique is referred to as SMOTE. By inserting a point between existing observations from the original dataset, this approach creates new observations. The K-Nearest Neighbors approach is used. SMOTE creates new minority instances by combining minority instances that already exist. For the minority class, it creates virtual training records using linear interpolation. For each example in the minority class, one or more of the k-nearest neighbors are randomly chosen to serve as these synthetic training records. Following the oversampling procedure, the data is rebuilt and can be subjected to several categorization models.



Oversampling Technique (SMOTE)

Splitting the features and target

```
[ ] x = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values
```

```
[ ] from imblearn.over_sampling import SMOTE
    smote = SMOTE(random_state=39)
    non_fraud_over, fraud_over = smote.fit_resample(x, y)
```

```
[ ] non_fraud_over_df = pd.DataFrame(non_fraud_over, columns=["distance_from_home", "distance_from_last_transaction",
           "ratio_to_median_purchase_price", "repeat_retailer", "used_chip",
           "used_pin_number", "online_order"])
    non_fraud_over_df
```

```
[ ] non_fraud_over_df["fraud"] = fraud_over
    df3 = non_fraud_over_df
```

```
[ ] print("df3 shape:", df3.shape)
    print(df3.info())
    df3.describe()
```

4-2-4 Correlation Analysis

A statistical technique called correlation analysis is used to determine the strength of any potential relationships between two variables or datasets.

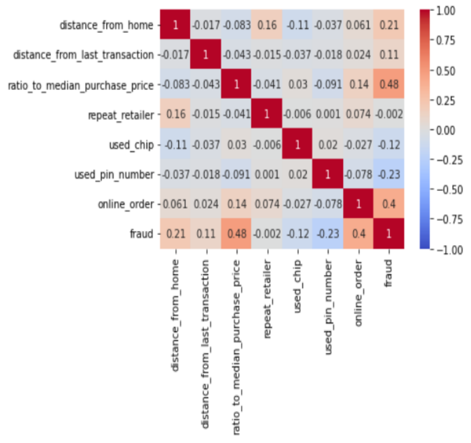


Fig 9: Heatmap for Correlation Analysis

As we can see here more positive the correlation more chances of fraud being caused. If the correlation is negative, fewer chances of fraud being caused.
Here we can see that ratio_to_median_purchase_price has a correlation of 0.48 with the fraud column. This means that there is a higher chance of fraud being caused because of the ratio_to_median_purchase_price.
As we can see, used_pin_number has a correlation of -0.23 with the fraud column. This means there are very less chances of fraud being caused because of used_pin_number.

--

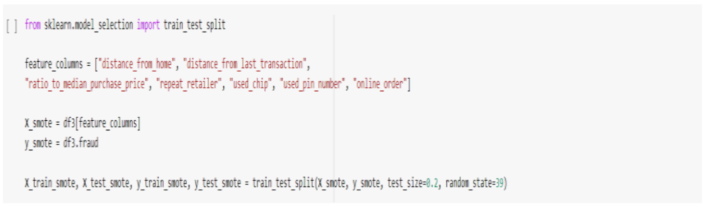
4-2-5 Training Phase

The dataset has 7 independent columns and 1 dependent column. The data set is divided into two subsets. The first subset, referred to as the training data, is a section of our actual dataset that is used to train a machine learning model.

You need unknown data to test your machine learning model after it has been constructed (using your training data). You can use this data, which is referred to as testing data, to assess the effectiveness and development of the training of your algorithms and to modify or optimize them for better outcomes.

Two main standards apply when testing data. It ought to:

- Display the actual dataset.
- Be ample enough to produce accurate predictions.



```
[ ] from sklearn.model_selection import train_test_split

    feature_columns = ["distance_from_home", "distance_from_last_transaction",
    "ratio_to_median_purchase_price", "repeat_retailer", "used_chip", "used_pin_number", "online_order"]

    X_smote = df3[feature_columns]
    y_smote = df3.fraud

    X_train_smote, X_test_smote, y_train_smote, y_test_smote = train_test_split(X_smote, y_smote, test_size=0.2, random_state=39)
```

x_smote refers to all the independent attributes whereas y_smote refers to the dependent attribute. From the above code we successfully split the data into training and test set data. 80% of the data set is the training data while 20% is the test data.

Algorithms:

We used five machine learning algorithms to find the accuracies of the model using the data set. Those 5 algorithms are KNN, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression.

1. **KNN Algorithm:**

A supervised machine learning (ML) technique known as K-nearest neighbors (KNN) can be applied to both classification and regression predicting issues. However, it is primarily employed in industry for classification predictive problems.

The two characteristics would accurately describe KNN:

- KNN is a lazy learning algorithm because it does not have a dedicated training phase and uses all of the data for training during classification.
- KNN is also a non-parametric learning algorithm because it makes no assumptions on the underlying data.

Working:

The K-nearest neighbors (KNN) method predicts the values of new datapoints based on "feature similarity," which further indicates that the new data point will be given a value depending on how closely it resembles the points in the training set. With the help of the processes below, we may comprehend how it functions.

Step 1: A dataset is necessary before we can build any algorithm. We must therefore load both the training and test data at the first step of KNN.

Step 2: The next step is to select the K value, or the closest data points. Any integer can be K.

Step 3: Complete the steps below for each point in the test data.

Calculate the separation between each row of training data and the test data.

2. **Decision Tree:**

A non-parametric supervised learning technique for classification and regression is called a decision tree (DT). The objective is to learn straightforward decision rules derived from the data features in order to build a model that predicts the value of a target variable.

Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node

Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes

Sub-tree – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

Pruning – is nothing but cutting down some nodes to stop overfitting.

Entropy:

Entropy is nothing but the uncertainty in our dataset or measure of disorder

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

Information Gain:

Information gain determines which attribute should be chosen as a decision node or root node by measuring the amount of uncertainty reduced by a given feature.

$$Information\ Gain = E(Y) - E(Y|X)$$

we will select the feature which has the highest information gain and then split the node based on that feature.

3. Random Forest

Like its name suggests, a random forest is made up of numerous independent decision trees that work together as an ensemble. Every tree in the random forest spits out a class forecast, and the classification that receives the most votes becomes the prediction made by our model.

The Random Forest Algorithm's ability to handle data sets with both continuous variables, as in regression, and categorical variables, as in classification, is one of its most crucial qualities. In terms of classification issues, it delivers superior outcomes.

Working:

We must first examine the ensemble technique in order to comprehend how the random forest algorithm in machine learning functions. Ensemble simply refers to the blending of various models. As a result, a group of models rather than a single model is employed to create predictions.

Ensemble uses two types of methods:

- **Bagging:**

  It replaces the sample training data with a different training subset, and the result is based on majority voting.
- **Boosting:**

  By building sequential models, it transforms poor learners into strong ones, with the final model having the best accuracy.

Random Forest works on the bagging principle. The ensemble method employed by random forest is bagging, sometimes referred to as Bootstrap Aggregation. A random sample is chosen from the data set using bagging. As a result, each model is created using the samples (Bootstrap Samples) that the Original Data gave, with a replacement process known as row sampling. Bootstrap refers to this stage of row sampling with replacement. Each model is currently trained independently, producing results. After merging the outputs of all the models, the final decision is made based on a majority vote. Aggregation is the process of aggregating all the results and producing a result based on a majority vote.

Steps involved in random forest algorithm:

Step 1: From a data set with k records, n random records are selected at random and used in the Random Forest algorithm.

Step 2: For each sample, a unique decision tree is built.

Step 3: An output will be produced by each decision tree.

Step 4: For classification and regression, the result is evaluated using a majority vote or an average.

4. Naïve Bayes :

The majority of large datasets utilize this model since it is simple to construct. For classification issues, a probabilistic machine learning model is used. The Bayes theorem and the assumption of predictor independence form the basis of the classifier. This implies that altering the value of one attribute does not alter the value of another.

Considering that it uses a probabilistic method, predictions can be made quite quickly. Both binary and multi-class classification issues can be solved with it. Understanding "Conditional Probability," "Bayes' Theorem," and how conditional probability aids in Bayes' Theorem is necessary before delving further into this subject.

Conditional Probability:

The possibility of an event or outcome happening contingent on the occurrence of a prior event or outcome is known as conditional probability. The probability of the prior event is multiplied by the current likelihood of the subsequent, or conditional, occurrence to determine the conditional probability.

Baye’s Rule:

We are trying to find the probability of event A, given event B is true.

Here P(B) is called prior probability which means it is the probability of an event before the evidence

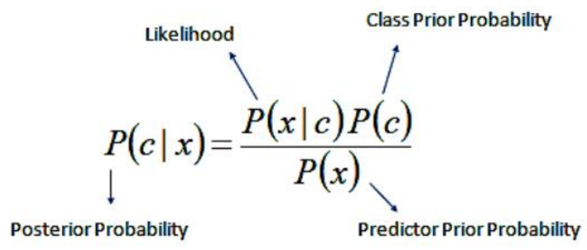
P(B|A) is called the posterior probability i.e., Probability of an event after the evidence is seen.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

When the features are independent, we can expand Bayes' rule to a method known as Naive Bayes, which makes the assumption that the features are independent, meaning that changing the value of one feature has no bearing on the values of other variables

$$P(Y=k|X) = \frac{P(X|Y=k) * P(Y=k)}{P(X)}$$

When there are multiple X variables, we simplify it by assuming that X's are independent.



$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|\mathbf{X}) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Assumptions of Naive Bayes

· All the variables are independent. That is if the animal is Dog that doesn't mean that Size will be Medium

· All the predictors have an equal effect on the outcome. That is, the animal being dog does not have more importance in deciding If we can pet him or not. All the features have equal importance.

5. ***Logistic Regression:*** The "Supervised machine learning" algorithm of logistic regression can be used to model the likelihood of a particular class or occurrence. It is applied when the outcome is binary or dichotomous and the data may be linearly separated.

That means problems involving binary classification are typically solved using logistic regression.

Predicting the output variable that is discrete in two classes is known as binary classification.

Binary classifications include Yes/No, Pass/Fail, Win/Lose, Cancerous/Non-cancerous, and many others.

Types of Logistic Regression:

- **Simple Logistic Regression:** a single independent is used to predict the output
- **Multiple logistic regression:** multiple independent variables are used to predict the output

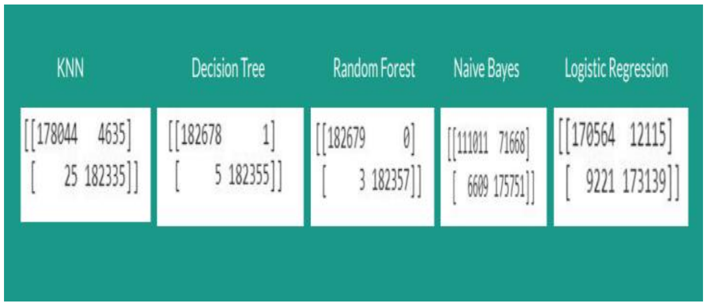
4-3 Testing Phase

Confusion Matrix:

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

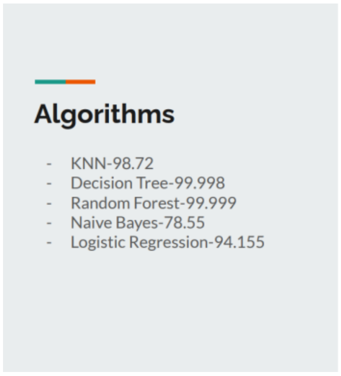
- **True Positives (TP):** These are cases in which we predicted yes, and the result is also yes.
- **True Negatives (TN):** We predicted no, and the result is also no.
- **False Positives (FP):** We predicted yes, but the result is no (Also known as a "Type I error.")
- **False Negatives (FN):** We predicted no, but the result is yes. (Also known as a "Type II error.")

The confusion matrix for all the five algorithms is as shown below:



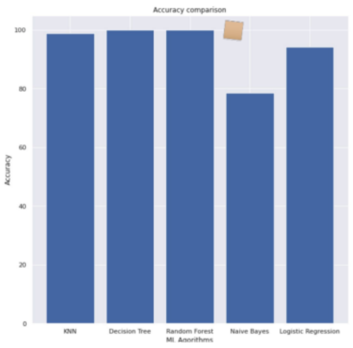
KNN	Decision Tree	Random Forest	Naive Bayes	Logistic Regression
[[178044 4635]	[[182678 1]	[[182679 0]	[[111011 71668]	[[170564 12115]
[25 182335]]	[5 182355]]	[3 182357]]	[6609 175751]]	[9221 173139]]

The accuracies of all the classifiers are as shown below:



Algorithms

- KNN-98.72
- Decision Tree-99.998
- Random Forest-99.999
- Naive Bayes-78.55
- Logistic Regression-94.155



We see that the Random Forest classifier is most accurate with the rate of 99.999% when compared to other classifiers.

5-Threats to Machine Learning Systems

5-1 Types of Threats to Machine Learning Systems

Depending on how well the learning model has been trained, there are two different sorts of security dangers to machine learning systems. Threats before or during model training and Threats after the machine learning model has been trained are the two types mentioned.

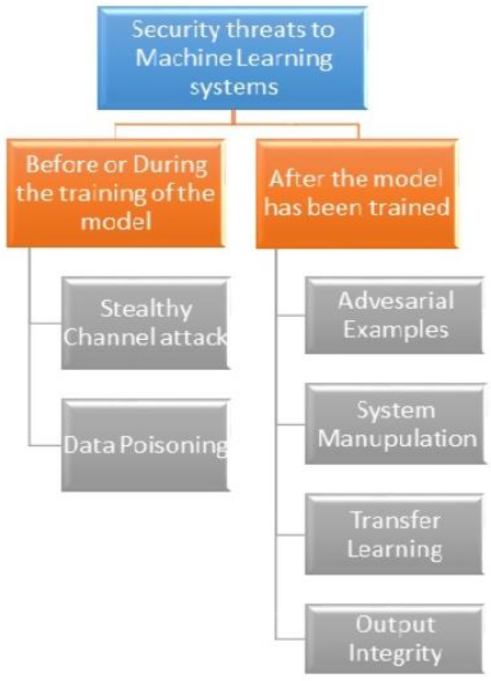

Security threats to Machine Learning systems
Before or During the training of the model
After the model has been trained
Stealthy Channel attack
Data Poisoning
Adversarial Examples
System Manupulation
Transfer Learning
Output Integrity

Fig 10: Types of Threats to Machine Learning Systems

1. **Stealthy Channel Attack**

   Attackers that wish to do harm search for minute differences in the model data inputs that may lead to redirected and undesirable model outputs. This is known as an adversarial machine learning attack. By altering just one pixel in the input image, for instance, a hacker can cause an image classification neural network to make an inaccurate forecast. This increases the likelihood that the neural network will incorrectly identify the image.

2. **Data Poisoning Attack**

   In a data poisoning attack, the attacker locates the source of the raw data that was used to train the model and then tries to skew or "poison" the data in order to reduce the accuracy of the machine learning model that results. The hacker needs to be inside or use other strategies to target employees in order to acquire access since they require access to the raw data.

3. **Transfer Learning Attack**

   Most models used in transfer learning are built using solution architecture that has been pre-trained and is based on a bigger data set. This approach has a lot of advantages, but it also leaves room for a backdoor transfer learning attack. It is likely that an attack will succeed on the existing model if it is practiced on the fundamental foundation upon which the model is formed.

5-2 Implementation of Data Poisoning Attack

In this attack, we performed a data poisoning attack against a Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel.

In this attack, we performed a data poisoning attack against a Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel.

We will first train the classifier, by evaluating its accuracy when it is not under attack.

We further divided the training set into two because the poisoning attack will also require a validation set to confirm the classifier's performance throughout the attack.

```
n_tr = 300  # Number of training set samples
n_val = 300  # Number of validation set samples
n_ts = 300 # Number of test set samples
```

Here we have taken a total of 900 samples, which are split into 300 each for training, validation, and testing set.

To compute a poisoning point, it is necessary to solve a bi-level optimization problem by:

$$\max_{x_c} A(D_{val}, \mathbf{w}^*) = \sum_{j=1}^{m} \ell(y_j, \mathbf{x}_j, \mathbf{w}^*)$$
$$s.t.\mathbf{w}^* \in \underset{\mathbf{w}}{\operatorname{argmin}} L(D_{tr} \cup (\mathbf{x}_c, y_c), \mathbf{w})$$

Here,
Xc is the poisoning point
A is the attacker objective function
L is the classifier training function
Dtr is the validation dataset
Dval is the training dataset

The training of the classifier is done with the poisoning point Xc on the poisoned data, and the latter of used to assess the accuracy of the untouched data.

Different CAttackPoisoning subclasses in SecML implement this attack. We employ the CAttackPoisoningSVM class to assault an SVM classifier.

```
Initial poisoning sample features: CArray([0.543615 0.761765]
Initial poisoning sample label: 1
```

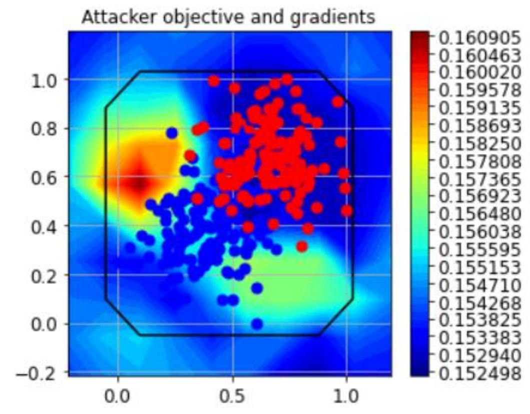


Fig 11: Visualizing a single poisoning point

We set the desired number of poisoning points to be generated equal to 20.

```
n_poisoning_points = 20  # Number of poisoning points to generate
pois_attack.n_points = n_poisoning_points
```

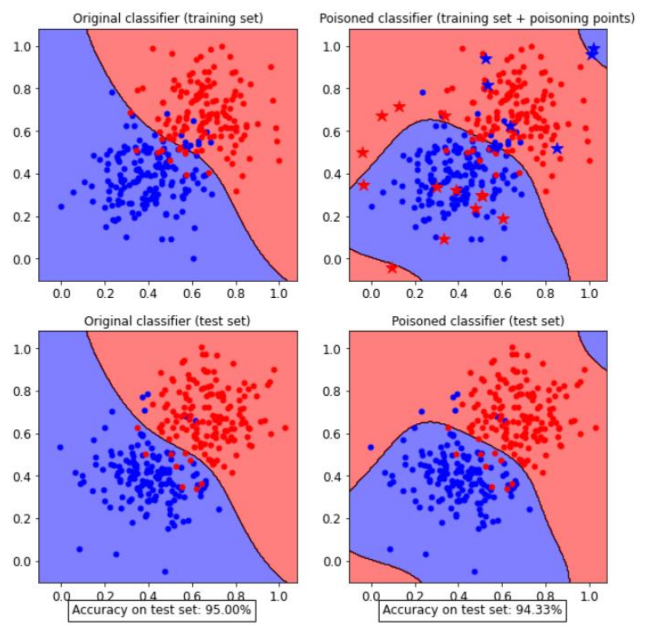


Fig 12: Accuracy after Data poisoning

As we can see before the attack the original classifier had an accuracy of 95.00%. After the data was poisoned the SVM decision boundary also got changed and the accuracy on the test set got reduced to 94.33%.

Hence we successfully implemented a data poisoning attack while training the model.

6-Conclusion

We have implemented 5 machine learning algorithms for detecting a fraud and have obtained a highest accuracy of 99.999%. This model can help the users after putting their transaction information to check whether will cause a fraud or not. We have also successfully implemented a data poisoning attack which decreases the accuracy of the SVM classifier from 95% to 94.33%.

7-References

1. https://www.analyticsvidhya.com/blog/2021/01/security-threats-to-machine-learning-systems/
2. https://www.linode.com/docs/guides/machine-learning-cyber-attacks/
3. https://www.excella.com/insights/ml-model-security-preventing-the-6-most-common-attacks
4. https://secml.readthedocs.io/en/stable/tutorials/05-Poisoning.html