**FRAUD DETECTION EFFECTIVENESS WITH CREDIT CARDS**

**AGENDA:**

Abstract

Introduction

Diagram

Software - Libraries Used

Algorithms

Conclusion

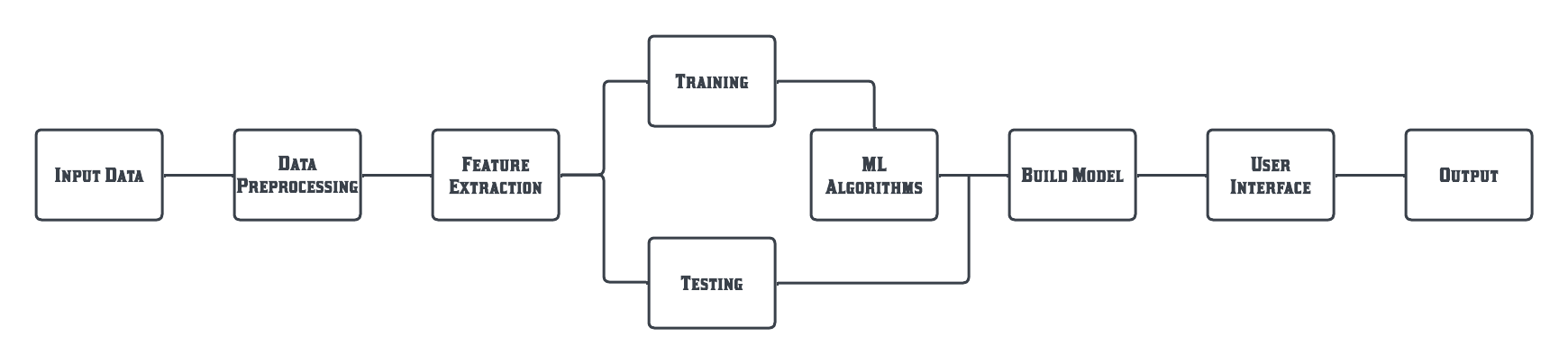
**ABSTRACT:**

The purpose of this project is to detect the fraudulent transactions made by credit cards by the use of machine learning techniques, to stop fraudsters from the unauthorized usage of customers’ accounts. The increase of credit card fraud is growing rapidly worldwide, which is the reason actions should be taken to stop fraudsters. Putting a limit for those actions would have a positive impact on the customers as their money would be recovered and retrieved back into their accounts and they won’t be charged for items or services that were not purchased by them which is the main goal of the project. Detection of the fraudulent transactions will be made by using three machine learning techniques SVM and Logistic Regression,Decision Tree those models will be used on a credit card transaction dataset.

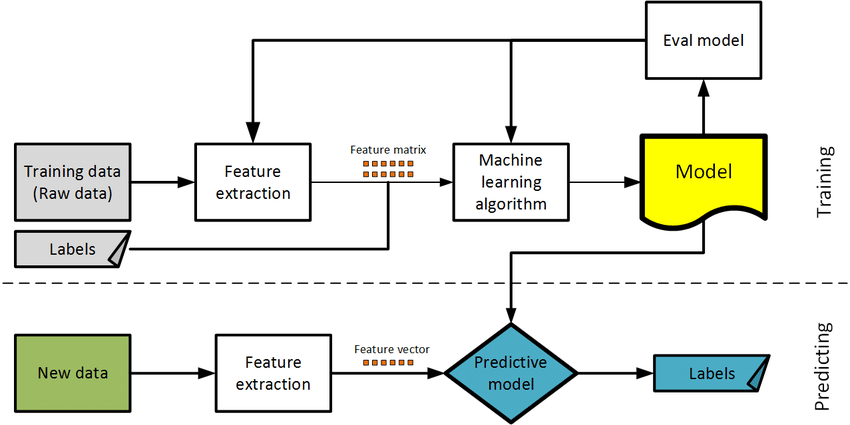
**INTRODUCTION:**

In the modern world, credit cards have revolutionized the way we conduct financial transactions, offering unparalleled convenience and flexibility. However, this convenience comes with a downside – the ever-looming threat of credit card fraud. Credit card fraud involves unauthorized or fraudulent transactions, and it poses a substantial risk to individuals, businesses, and the financial institutions that underpin our economic system.The purpose of this project is to address this critical issue by developing a comprehensive credit card fraud detection system. This system leverages cutting-edge machine learning algorithms in combination with the Flask web framework, aiming to significantly enhance the effectiveness of fraud detection processes. Furthermore, it provides an intuitive and user-friendly interface for seamless interaction with the system. Credit card fraud detection is not merely a financial concern; it is a crucial aspect of preserving the trust and reliability of electronic payment systems. Effective fraud detection ensures the protection of individuals' financial assets and minimizes disruptions to their lives. It also plays a pivotal role in maintaining the integrity of the broader financial industry, where transactions worth trillions of dollars occur daily. The subsequent sections of this project will delve into the methodologies employed, the specific objectives, and the anticipated outcomes. This project underscores the paramount importance of having robust fraud detection mechanisms in place in today's digitally connected financial landscape.

**BLOCK DIAGRAM:**



**FLOW DIAGRAM:**

****

**UML DIAGRAM**

Unified Modelling Language (UML) is simply another graphical

representation of a common semantic model. The proposed system has been

designed by using use case diagram, class diagram, sequence diagram,

collaboration diagram, state chart diagram and component diagram.

**USE CASE DIAGRAM**

The use case diagram consists of the actors and the use cases. The actors

of the system are user, system holder, device controller and the use cases are authentication, checking credentials, basic ON/OFF, allow/deny user, storing NLP commands, Input through voice commands, Deriving Data, Intrusion Detection, Service Maintenance. Figure No. 3.2 describes Use Case diagram for Adaptive Automation System (AAS).

**MODULES:**

**Data Collection**: Machine learning models require data as their input. This data can come in various forms, such as text, images, numerical values, or a combination of these. High-quality and representative data are essential for training an effective model.

**Data Preprocessing**: Raw data often requires preprocessing to clean, normalize, and prepare it for training. This may involve tasks like removing outliers, handling missing values, and scaling numerical features.

**Feature Extraction**: In some cases, feature engineering is performed to extract relevant information from the data or create new features that can improve the model's performance.

**Model Selection**: There are various types of machine learning models, including supervised (e.g., decision trees, neural networks), unsupervised (e.g., clustering algorithms), and reinforcement learning models. The choice of the model depends on the specific task and the characteristics of the data.

**Model Training**: During the training phase, the model is fed with labeled data (in supervised learning) or unlabeled data (in unsupervised learning). The model iteratively adjusts its internal parameters to minimize a predefined loss function, effectively learning to make predictions or discover patterns.

**Training Data**

Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict. If you are using supervised learning or some hybrid that includes that approach, your data will be enriched with data labeling or annotation.

**Testing Data**

Test data is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine. Test data will help you see how well your model can predict new answers, based on its training. Both training and test data are important for improving and validating machine learning models.

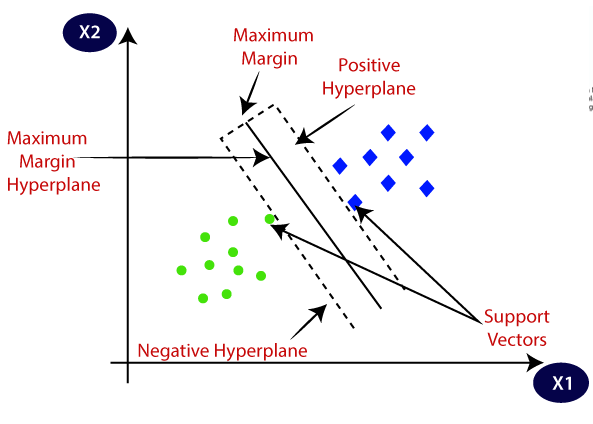
**Deployment**: Once the model performs satisfactorily, it can be deployed in real-world applications. Deployment may involve integrating the model into software systems, web applications, or other environments where it can make predictions or assist in decision-making.

**ALGORITHM USED:**

* Support Vector Machines
* Decision Tree
* Logistic Regression

**Support Vector Machines**

Machine learning techniques can categorize plant diseases, with the Support Vector Machine (SVM) being a popular choice due to its effectiveness in high dimensional spaces and memory efficiency. SVM is a discriminative classifier that uses a hyperplane to distinguish between different groups of scattered data points. The classification process involves selecting whether the input image is healthy or diseased, with the hyperplane being the most effective method for identifying two classes. This method is particularly useful in situations where the number of dimensions exceeds the number of samples.



**SUPPORT VECTORS:**

Datapoints that are closes to the hyperplane is called support vector. Separating line will be defined with the help of these data points.

**HYPERPLANE**:

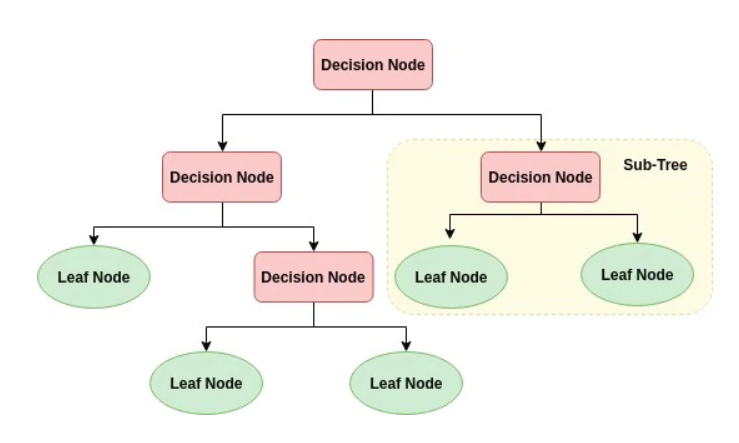
It is a decision plane or space which is divided between a set of objects having different classes.

**MARGIN:**

It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

**DECISION TREE ALGORITHM:**

A decision tree is a popular machine learning algorithm used for both classification and regression tasks. It's a tree-like structure where each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf node represents the outcome or class label. Decision trees are simple to understand and interpret, making them useful for various applications.



**DT algorithm worked:**

1. **Start at the Root Node**:
   1. The root node is the topmost node of the decision tree, containing all the data.
2. **Select the Best Feature**:
   1. The algorithm evaluates each feature in the dataset to find the one that best splits the data into more homogeneous subsets. It does this by calculating a measure of impurity, such as Gini impurity or entropy, for different possible splits.
   2. The feature that results in the greatest reduction in impurity is selected as the best feature to split on.
3. **Create a Decision Node**:
   1. The chosen feature becomes a decision node (internal node) in the tree.
4. **Split the Data**:
   1. The data is divided into subsets based on the values of the chosen feature. Each subset represents a branch from the decision node.
   2. For example, if the feature is "age" and the decision node tests whether age is less than 30, there will be two branches: one for age < 30 and one for age ≥ 30.

**5.Repeat for Each Subset**:

* 1. The above steps are recursively applied to each subset created by the split.
  2. The algorithm evaluates the features within each subset to find the best feature for the next split, and the process continues.

**6. Stopping Criteria**:

* 1. The recursive splitting process continues until one of the stopping criteria is met. Common stopping criteria include:
     1. Maximum tree depth: The tree cannot exceed a certain depth.
     2. Minimum number of samples per leaf: A node cannot be split if it has fewer than a certain number of data points.
     3. No further reduction in impurity: If further splits do not significantly reduce impurity, the splitting process stops.

**7. Assign Class Labels or Values:**

* + Once the stopping criteria are met, the leaf nodes contain subsets of data that are relatively homogeneous with respect to the target variable (for classification or regression).
  + For classification problems, each leaf node is assigned the class label that is most prevalent among the data points in that node.
  + For regression problems, each leaf node is assigned the average or median value of the target variable within that node.

**8. Predictions:**

* + To make predictions for new data, start at the root node and traverse the tree by following the decision rules at each internal node based on the feature values of the new data.
  + Eventually, you will reach a leaf node, and the class label (for classification) or predicted value (for regression) of that leaf node becomes the prediction for the new data point.

**9.Evaluation and Pruning** :

* + You can evaluate the performance of the decision tree using metrics like accuracy (for classification) or mean squared error (for regression) on a validation dataset.
  + Pruning techniques may be applied to simplify the tree by removing nodes that do not significantly improve performance, helping to prevent overfitting.

**10. Deployment**:

* + Once you are satisfied with the decision tree's performance, you can deploy it for making predictions on new, unseen data.

**SOFTWARE - LIBRARIES USED:**

Language : Python

Tool : VS Code(Python)

Splitting Training & Testing: sklearn.model\_selection.train\_test\_split

Files Read : Pandas

Mathematics : Numpy & Math

Graph : Matplotlib & Seaborn

Dimensionality Reduction : PCA

Dataset Balancing : imblearn.over\_sampling.SMOTE

Cross Validation : GridSearchCV

Standardisation : sklearn.preprocessing.StandardScaler

Score : accuracy\_score, classification\_report, confusion\_matrix,

Algorithms:

* Logistic Regression
* SVM - SVC
* Decision Tree

**ABOUT MACHINE LEARNING:**

**UML DIAGRAM**

Unified Modelling Language (UML) is simply another graphical

representation of a common semantic model. The proposed system has been

designed by using use case diagram, class diagram, sequence diagram,

collaboration diagram, state chart diagram and component diagram.

**USE CASE DIAGRAM**

The use case diagram consists of the actors and the use cases. The actors

of the system are user, system holder, device controller and the use cases are authentication, checking credentials, basic ON/OFF, allow/deny user, storing NLP commands, Input through voice commands, Deriving Data, Intrusion Detection, Service Maintenance. Figure No. 3.2 describes Use Case diagram for Adaptive Automation System (AAS).

User and Non-User can login to the device by Admin. Then device controller decides the entry of the user. The user can control the devices through voice commands and these can be monitored by Device Controller. Intrusion detection and maintenance also controlled by the Device Controller (Admin). The user can check for the intruder in the environment and based on the intruder the door can be open/closed. The light can be turned on/off by the system holder and they can alert the emergency system if any poisonous gas or temperature exceeds the normal level. This can be maintained by system maintenance.

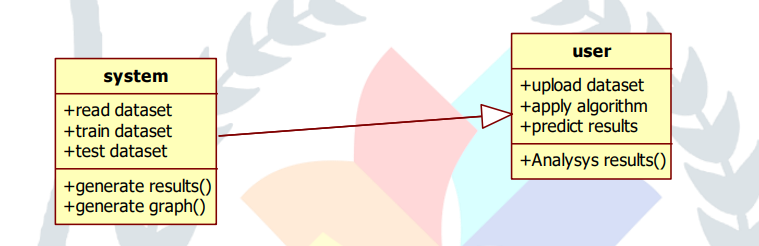


**CLASS DIAGRAM**

Class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction and it is used for general conceptual modelling of the structure of the application, and for detailed modelling translating the models into programming code.

Class diagrams can also be used for data modelling. The classes in a class

diagram represent both the main elements, interactions in the application, and the classes to be programmed.



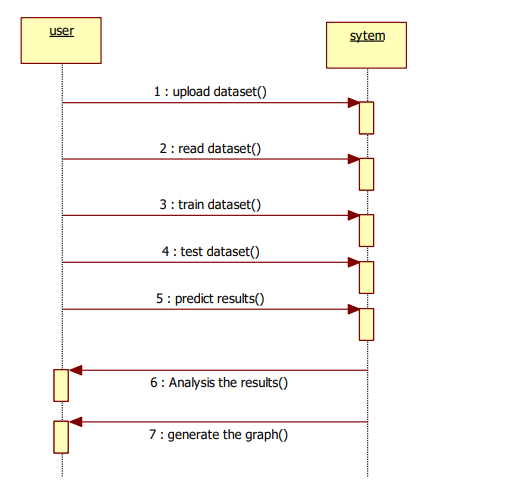
**SEQUENCE DIAGRAM**

The control flow between various participants or entity roles of the

corresponding system in the form of messages is represented in the Sequence Diagram. The participants are represented within the rectangular object. The swim line or the lifeline that is dragged below every participant represents the lifetime of the corresponding participant.

The UML representation of a class is rectangle containing three

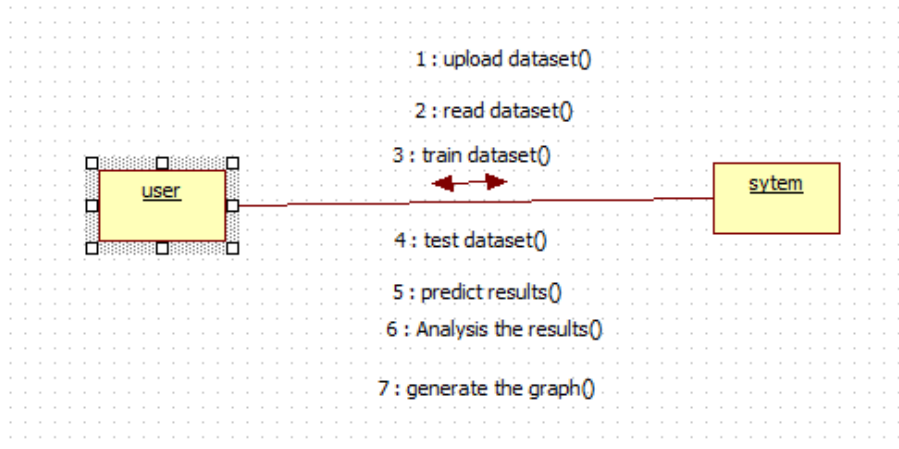
compartments stacked vertically. The top compartments shows the class’s name. the middle compartments list the class’s attributes. The bottom compartment lists the class operations known as the methods of the class. A class diagram consists of any number of classes which will be connected by the lines, which may have arrows at one or both ends, connecting the boxes. These lines define the relationships, also called associations, between the classes. These lines will have multiplicity to represent the number of instances of the classes.



**COLLABORATION DIAGRAM**

Collaboration diagram is defined as one of the interaction diagram,

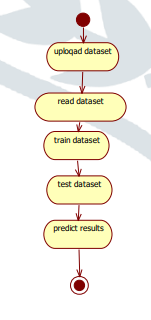
which consists of the set of objects related in a particular context and interaction among those objects. The collaboration diagram is also called as the set of message exchange among the objects within the collaborative nature of message exchange between the corresponding objects.



**ACTIVITY DIAGRAM**

Activity diagram is another important diagram in UML to describe the

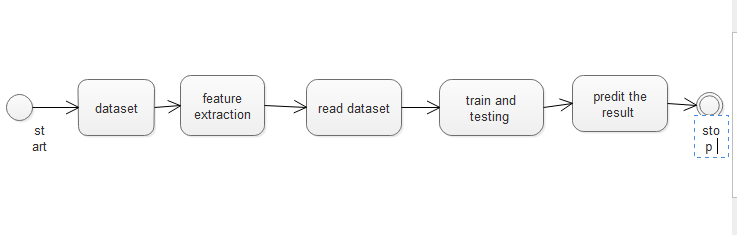
dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another with different components of activity diagram. Some of the components of activity diagram Start/Stop symbol, Action symbol, Joint and Fork symbol, Decision symbol, Connector symbol.



**STATE CHART DIAGRAM**

State chart diagram is one of the five UML diagrams used to model the

dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. There are three main components in the State chart diagram such as Initial/ Final states, State symbol and Transition symbol



**CONCLUSION:**

In conclusion, the main objective of this project was to find the most suited model in credit card fraud detection in terms of the machine learning techniques chosen for the project, and it was met by building the four models and finding the accuracies of them all, the best model in terms of accuracies is Support Vector Machine which scored 99.94% with only 51 misclassified instances. I believe that using the model will help in decreasing the amount of credit card fraud and increase the customers satisfaction as it will provide them with better experience in addition to feeling secure.