**Credit Card Fraud Detection: A Comparative Study of Machine Learning Algorithms on Imbalanced Data**

**ABSTRACT**

Credit card fraud refers to the physical loss of credit card or loss of sensitive credit card information. Many machine learning algorithms can be used for detection. This research shows several algorithms that can be used for classifying transactions as fraud or genuine one. Credit Card Fraud Detection dataset was used in the research. Because the dataset was highly imbalanced, SMOTE technique was used for oversampling. Further, feature selection was performed and dataset was split into two parts, training data and test data. The algorithms used in the experiment were Logistic Regression, Random Forest, and Decision Tree. Results show that each algorithm can be used for credit card fraud detection with high accuracy. Proposed model can be used for detection of other irregularities.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **LIST OF TABLES** | i  v  vii  xi  xii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.3.1EXISTINGSYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.2.4GIVEN INPUTAND EXPECTED OUTPUT  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 : SYSTEM DESIGN**  **4.1 general**  4.1.1 activity diagram  4.1.2 USE CASE DIAGRAM  4.1.3 DATA FLOW DIAGRAM  4.1.4SEQUENCE DIAGRAM  4.1.5 COLLABORATION DIAGRAM  4.1.6CLASS DIAGRAM  4.1.7 SYSTEM ARCHITECTURE  4.1.8 OBJECT DIAGRAM  4.1.9 STATE DIAGRAM  4.1.10 COMPONENT DIAGRAM  4.1.11 E-R DIAGRAM  4.2 DATABASE DESIGN (ALL LEVEL |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 Python  5.2 History of python  5.3 impartance of python  5.4 features of python  5.5 libraries used |  |
| 6. | **CHAPTER 6 : IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |
| 7. | **CHAPTER 7 : SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 : SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS |  |
| 10. | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.1 | Use case Diagram |  |
| 4.2 | Class Diagram |  |
| 4.3 | Object Diagram |  |
| 4.4 | Component Diagram |  |
| 4.5 | Deployment Diagram |  |
| 4.6 | Sequence diagram |  |
| 4.7 | Collaboration diagram |  |
| 4.8 | State Diagram |  |
| 4.9 | Activity Diagram |  |
| 4.10 | Data flow diagram |  |
| 4.11 | E-R Diagram |  |
| 4.12 | Architecture Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | ML | Machine Learning |
| 2. | SVM | Support Vector Machine |
| 3. | CNN | Convolutional Neural Networks |
| 4. | ANN | Artificial Neural Networks |
| 5. | AI | Artificial Intelligence |
| 6. | DNN | Deep Neural Networks |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

There are growing number of new companies all around the world. All of that companies are trying to provide best service quality for their customers. In order to succeed in that, companies are processing a lot of data on a daily basis. These data come from vast number of resources and are in different formats. Moreover, this data contains some of the key parts of the company’s future business. Because of that, companies need to store that data, to process it and what is really important, to keep it safe. Without securing data, a lot of it can be used by other companies or even worse, it can be stolen. In most cases, financial information is stolen, which can harm whole company or individual.

**1.2 OBJECTIVE**

The objective of this paper is to analyze various machine-learning algorithms, such as Logistic Regression (LR), Random Forest (RF), and Decision Tree (DT) in order to determine which algorithm is most suitable for credit card fraud detection.

**1.3 Existing System:**

Check Fraud occurs when person forges a check or pays for something with check knowing that there is not enough money. Internet sales are fraud where fraudster sale fake items or counterfeit items, or taking payment without delivering the item. There are a couple more, such as charities fraud, identity theft, credit card fraud, debt elimination, Insurance fraud and others. Due to increasing popularity of cashless transactions, one of the most common frauds are credit card frauds. Credit card fraud refers to the situation where fraudster uses credit card for their needs while owner of that credit card is not aware of that.

**Disadvantages:**

* Less accurate.

**LITERATURE SURVEY:**

**Title:** Credit Card Fraud Detection on the Skewed Data Using Various Classification and Ensemble Techniques

**Author:** Ankit Mishra

**Year:** 2018

**Description:**

Nowadays, as internet speed has increased and the prices of mobile have decreased very much in past few years. Also, the data prices too are very much affordable to most of the people. This has resulted into the digitization of most of the institutes as it is easy and convenient for the people and also for the authority to maintain the records. So, it resulted in most of the banks and other institutes receiving and transferring money through credit cards. But with the hackers and other cyber criminals around there is always chances of the frauds in the transactions. The possibility of the fraud transaction is very less but it is not negligible and even having one fraud transaction is unacceptable because it is crime and we can't neglect it even if it is very less as it harms both the customer and credibility of the institute. So this paper aims at analyzing various classification techniques using various metrics for judging various classifiers. This model aims at improving fraud detection rather than misclassifying a genuine transaction as fraud.

**Title:** Machine Learning for Credit Card Fraud Detection System

**Author:** Lakshmi S V S S

**Year:** 2018

**Description:**

The rapid growth in E-Commerce industry has lead to an exponential increase in the use of credit cards for online purchases and consequently they has been surge in the fraud related to it .In recent years, For banks has become very difficult for detecting the fraud in credit card system. Machine learning plays a vital role for detecting the credit card fraud in the transactions. For predicting these transactions banks make use of various machine learning methodologies, past data has been collected and new features are been used for enhancing the predictive power. The performance of fraud detecting in credit card transactions is greatly affected by the sampling approach on data-set, selection of variables and detection techniques used. This paper investigates the performance of logistic regression, decision tree and random forest for credit card fraud detection. Dataset of credit card transactions is collected from kaggle and it contains a total of 2,84,808 credit card transactions of a European bank data set. It considers fraud transactions as the “positive class” and genuine ones as the “negative class”. The data set is highly imbalanced, it has about 0.172% of fraud transactions and the rest are genuine transactions. The author has been done oversampling to balance the data set, which resulted in 60% of fraud transactions and 40% genuine ones. The three techniques are applied for the dataset and work is implemented in R language. The performance of the techniques is evaluated for different variables based on sensitivity, specificity, accuracy and error rate. The result shows of accuracy for logistic regression, Decision tree and random forest classifier are 90.0, 94.3, 95.5 respectively. The comparative results show that the Random forest performs better than the logistic regression and decision tree techniques.

**Title:** Analysis on Credit Card Fraud Identification Techniques based on KNN and Outlier Detection

**Author:** N. Malini

**Year:** 2017

**Description:**

Popular payment mode accepted both offline and online is credit card that provides cashless transaction. It is easy, convenient and trendy to make payments and other transactions. Credit card fraud is also growing along with the development in technology. It can also be said that economic fraud is drastically increasing in the global communication improvement. It is being recorded every year that the loss due to these fraudulent acts is billions of dollars. These activities are carried out so elegantly so it is similar to genuine transactions. Hence simple pattern related techniques and other less complex methods are really not going to work. Having an efficient method of fraud detection has become a need for all banks in order to minimize chaos and bring order in place. There are several techniques like Machine learning, Genetic Programming, fuzzy logic, sequence alignment, etc are used for detecting credit card fraudulent transactions. Along with these techniques, KNN algorithm and outlier detection methods are implemented to optimize the best solution for the fraud detection problem. These approaches are proved to minimize the false alarm rates and increase the fraud detection rate. Any of these methods can be implemented on bank credit card fraud detection system, to detect and prevent the fraudulent transaction.

**Title:** Credit Card Nearest Neighbor Based Outlier Detection Techniques

**Author:** Mrs.C.Navamani

**Year:** 2018

**Description:**

Popular payment mode accepted both offline and online is credit card that provides cashless transaction. It is easy, convenient and trendy to make payments and other transactions. Credit card fraud is also growing along with the development in technology. It can also be said that economic fraud is drastically increasing in the global communication improvement. It is being recorded every year that the loss due to these fraudulent acts is billions of dollars. These activities are carried out so elegantly so it is similar to genuine transactions. Hence simple pattern related techniques and other less complex methods are really not going to work. Having an efficient method of fraud detection has become a need for all banks in order to minimize chaos and bring order in place. There are several techniques like Machine learning, Genetic Programming, fuzzy logic, sequence alignment, etc are used for detecting credit card fraudulent transactions. Along with these techniques, KNN algorithm and outlier detection methods are implemented to optimize the best solution for the fraud detection problem. These approaches are proved to minimize the false alarm rates and increase the fraud detection rate. Any of these methods can be implemented on bank credit card fraud detection system, to detect and prevent the fraudulent transaction.

**Title:** Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis

**Author:** John O. Awoyemi

**Year:** 2017

**Description:**

Financial fraud is an ever growing menace with far consequences in the financial industry. Data mining had played an imperative role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a data mining problem, becomes challenging due to two major reasons - first, the profiles of normal and fraudulent behaviours change constantly and secondly, credit card fraud data sets are highly skewed. The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on dataset, selection of variables and detection technique(s) used. This paper investigates the performance of naïve bayes, k-nearest neighbor and logistic regression on highly skewed credit card fraud data. Dataset of credit card transactions is sourced from European cardholders containing 284,807 transactions. A hybrid technique of under-sampling and oversampling is carried out on the skewed data. The three techniques are applied on the raw and preprocessed data. The work is implemented in Python. The performance of the techniques is evaluated based on accuracy, sensitivity, specificity, precision, Matthew’s correlation coefficient and balanced classification rate. The results show of optimal accuracy for naïve bayes, k-nearest neighbor and logistic regression classifiers are 97.92%, 97.69% and 54.86% respectively. The comparative results show that k-nearest neighbor performs better than naïve bayes and logistic regression techniques.

* 1. **Proposed System**

There are two types of credit card frauds. One is theft of physical card, and other one is stealing sensitive information from the card, such as card number, cvv code, type of card and other. By stealing credit card information, a fraudster can broach a large amount of money or make a large amount of purchase before cardholder finds out. Because of that, companies use various machine learning methods to recognize which transactions are fraudulent and which are not.

**ADVANTAGES**

* More Accurate

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Experiment included back propagation neural network that was optimized with Whale algorithm. Neural network consisted of 2 input layers, 20 hidden and 2 output layers. Due to optimization algorithm, they achieved exceptional results on 500 test samples: 96.40% accuracy and 97.83% recall. Authors of paper and used neural networks, in order to demonstrate improvement in results when ensemble techniques are used. In paper three datasets were used for comparison between Auto-encoder and Restricted Boltzmann Machine algorithms, which led to the conclusion that algorithms like MLP can be suitable for credit card fraud detection.

**2.2 METHODOLOGIES**

**2.2.1** **MODULES NAME:**

1. Data Gathering

2. Data Exploration

3. Data Manipulation

4. Data Modeling

5. Fitting

**Data Gathering:**

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. The goal for all data collection is to capture quality evidence that allows analysis to lead to the formulation of convincing and credible answers to the questions that have been posed. Here we need to gather the data which used for detecting the credit card frauds.

**Data Exploration:**

After gathering the data from web, we will explore the data that is contained in the credit card data frame. We will proceed by displaying the credit card data using the head function as well as the tail function. We will then proceed to explore the other components of this data frame.

**Data Manipulation:**

We will apply this to the amount component of our credit card data amount. Scaling is also known as feature standardization. With the help of scaling, the data is structured according to a specified range. Therefore, there are no extreme values in our dataset that might interfere with the functioning of our model.

**Data Modeling:**

Data modeling is the process of creating a data model for the data to be stored in a dataset. This data model is a conceptual representation of Data objects, the associations between different data objects and the rules. Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. We will split our dataset into training set as well as test set with a split ratio.

**Fitting:**

We will implement a decision tree algorithm. Decision Trees to plot the outcomes of a decision. These outcomes are basically a consequence through which we can conclude as to what class the object belongs to. We will now implement our decision tree model.

* 1. **TECHNIQUE USED OR ALGORITHM USED**

**Multilayer Perception**

* The purpose of this paper is to analyze various machine learning algorithms, such as Logistic Regression (LR), Random Forest (RF), Naïve Bayes (NB) and Multilayer Perceptron (MLP) in order to determine which algorithm is most suitable for credit card fraud detection.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

We can see from the results that on each database, the error rates are very low due to the discriminatory power of features and the regression capabilities of classifiers. Comparing the highest accuracies (corresponding to the lowest error rates) to those of previous works, our results are very competitive.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

**SOFTWARE REQUIREMENTS**

* Operating System : Windows 7/8/10
* Platform : Spyder3
* Programming Language : Python, HTML
* Front End : Spyder3

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering.

**UML Diagrams**

Use Case diagram



Fig 4.1: Use Case Diagram

**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**Class Diagram**



Fig 4.2: Class Diagram

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**Object Diagram**



Fig 4.3: Object Diagram

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**Component Diagram**



Fig 4.4: Component Diagram

**Deployment Diagram**



Fig 4.5: Deployment Diagram

**Sequence Diagram**



Fig 4.6: Sequence Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Collaboration Diagram**



Fig 4.7: Collaboration Diagram

**State Diagram**



Fig 4.8: State Diagram

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**Activity Diagram**



Fig 4.9: Activity Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Data Flow Diagram**

**Level 0**

Dataset

Preprocessing

Prediction & Accuracy

Splitting

Apply Algorithms

**Level 1**

Training Data, Testing Data

Input Data Set

Preprocessing

Diabetes Prediction

Data, Cleaning Data, Remove Null Values

Apply Random Forest, SVM, Decision Tree Algorithms

Splitting

Detection

Fig 4.10: Data Flow Diagrams

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**E-R Diagram**

Preprocessing

Splitting

CC Fraud

User

Algorithms

Fig 4.11: E-R Diagram

**EXPLANATION:**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**System Architecture**

Data Gathering

Data Exploration

Manipulation

Preprocessing

Data Splitting

Modeling

Apply Algo

Accuracy

Fig 4.12: System Architecture

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### 5.3 Importance of Python

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 5.4 Features of Python

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.5 Libraries used in python**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

# Importing the dataset

dataset = pd.read\_csv('../Dataset/diabetes.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 8].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Parameter evaluation

treeclf = DecisionTreeClassifier(random\_state=42)

parameters = {'max\_depth': [6, 7, 8, 9],

'min\_samples\_split': [2, 3, 4, 5],

'max\_features': [1, 2, 3, 4]

}

gridsearch=GridSearchCV(treeclf, parameters, cv=100, scoring='roc\_auc')

gridsearch.fit(X,y)

print(gridsearch.best\_params\_)

print(gridsearch.best\_score\_)

# Adjusting development threshold

tree = DecisionTreeClassifier(max\_depth = 6, max\_features = 4,

min\_samples\_split = 5,

random\_state=42)

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y, random\_state=42)

tree.fit(X\_train, y\_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X\_train, y\_train)))

print("Accuracy on test set: {:.3f}".format(tree.score(X\_test, y\_test)))

# Predicting the Test set results

y\_pred = tree.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('TP - True Negative {}'.format(cm[0,0]))

print('FP - False Positive {}'.format(cm[0,1]))

print('FN - False Negative {}'.format(cm[1,0]))

print('TP - True Positive {}'.format(cm[1,1]))

print('Accuracy Rate: {}'.format(np.divide(np.sum([cm[0,0],cm[1,1]]),np.sum(cm))))

print('Misclassification Rate: {}'.format(np.divide(np.sum([cm[0,1],cm[1,0]]),np.sum(cm))))

round(roc\_auc\_score(y\_test,y\_pred),5)

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

from sklearn.neighbors import KNeighborsClassifier

# Importing the dataset

dataset = pd.read\_csv('../Dataset/diabetes.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 8].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Parameter evaluation

knnclf = KNeighborsClassifier()

parameters={'n\_neighbors': range(1, 20)}

gridsearch=GridSearchCV(knnclf, parameters, cv=100, scoring='roc\_auc')

gridsearch.fit(X, y)

print(gridsearch.best\_params\_)

print(gridsearch.best\_score\_)

# Fitting K-NN to the Training set

knnClassifier = KNeighborsClassifier(n\_neighbors = 18)

knnClassifier.fit(X\_train, y\_train)

print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knnClassifier.score(X\_train, y\_train)))

print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knnClassifier.score(X\_test, y\_test)))

# Predicting the Test set results

y\_pred = knnClassifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('TP - True Negative {}'.format(cm[0,0]))

print('FP - False Positive {}'.format(cm[0,1]))

print('FN - False Negative {}'.format(cm[1,0]))

print('TP - True Positive {}'.format(cm[1,1]))

print('Accuracy Rate: {}'.format(np.divide(np.sum([cm[0,0],cm[1,1]]),np.sum(cm))))

print('Misclassification Rate: {}'.format(np.divide(np.sum([cm[0,1],cm[1,0]]),np.sum(cm))))

round(roc\_auc\_score(y\_test,y\_pred),5)

# Importing the libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

from sklearn.svm import SVC

import matplotlib.pyplot as plt

from sklearn.metrics import classification\_report

# Importing the dataset

dataset = pd.read\_csv('../Dataset/diabetes.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 8].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

#svm with grid search

svm = SVC(random\_state = 42)

parameters = {'kernel':('linear', 'rbf'), 'C':(1,0.25,0.5,0.75),

'gamma': (1,2,3,'auto'),'decision\_function\_shape':('ovo','ovr'),

'shrinking':(True,False)}

scores = ['precision', 'recall']

for score in scores:

print("# Tuning hyper-parameters for %s" % score)

print()

svm = GridSearchCV(SVC(), parameters, cv=5,

scoring='%s\_macro' % score)

svm.fit(X\_train, y\_train)

print("Best parameters set found on development set:")

print()

print(svm.best\_params\_)

print()

print("Grid scores on development set:")

print()

means = svm.cv\_results\_['mean\_test\_score']

stds = svm.cv\_results\_['std\_test\_score']

for mean, std, params in zip(means, stds, svm.cv\_results\_['params']):

print("%0.3f (+/-%0.03f) for %r"

% (mean, std \* 2, params))

print()

print("Detailed classification report:")

print()

print("The model is trained on the full development set.")

print("The scores are computed on the full evaluation set.")

print()

y\_true, y\_pred = y\_test, svm.predict(X\_test)

print(classification\_report(y\_true, y\_pred))

print()

svm\_model = SVC(kernel='rbf', C=100, gamma = 0.0001, random\_state=42)

svm\_model.fit(X\_train, y\_train)

spred = svm\_model.predict(X\_test)

print ('Accuracy with SVM {0}'.format(accuracy\_score(spred, y\_test) \* 100))

# Making the Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('TP - True Negative {}'.format(cm[0,0]))

print('FP - False Positive {}'.format(cm[0,1]))

print('FN - False Negative {}'.format(cm[1,0]))

print('TP - True Positive {}'.format(cm[1,1]))

print('Accuracy Rate: {}'.format(np.divide(np.sum([cm[0,0],cm[1,1]]),np.sum(cm))))

print('Misclassification Rate: {}'.format(np.divide(np.sum([cm[0,1],cm[1,0]]),np.sum(cm))))

svm.fit(X\_train, y\_train)

round(roc\_auc\_score(y\_test,y\_pred),5)

**CHAPTER 7**

**SNAPSHOTS**

**General:**

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**SNAPSHOTS**

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3Types of Tests**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**CONCLUSION & REFERENCE**

**9.1 CONCLUSION**

Credit card frauds represent a very serious business problem. These frauds can lead to huge losses, both business and personal. Because of that, companies invest more and more money in developing new ideas and ways that will help to detect and prevent frauds. The main goal of this paper was to compare certain machine learning algorithms for detection of fraudulent transactions. Hence, comparison was made and it was established that Random Forest algorithm gives the best results i.e. best classifies whether transactions are fraud or not. This was established using different metrics, such as recall, accuracy and precision. For this kind of problem, it is important to have recall with high value. Feature selection and balancing of the dataset have shown to be extremely important in achieving significant results.

**REFERENCES**

[1] Global Facts (2019). Topic: Startups worldwide. [online] Available at: https://www.statista.com/topics/4733/startups-worldwide/ [Accessed 10 Jan. 2019].

[2] Legal Dictionary (2019). Fraud - Definition, Meaning, Types, Examples of fraudulent activity. [online] Available at: https://legaldictionary.net/fraud/ [Accessed 15 Jan. 2019].

[3] European Central Bank (2018). Fifth report on card fraud, September 2018. [online]. Available at: https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport201809.en.html#toc1 [Accessed 21 Jan. 2019].

[4] En.wikipedia.org. (2019). Credit card fraud. [online] Available at: https://en.wikipedia.org/wiki/Credit\_card\_fraud [Accessed 24 Jan. 2019].

[5] A. Mishra, C. Ghorpade, “Credit Card Fraud Detection on the Skewed Data Using Various Classification and Ensemble Techniques” 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) pp. 1-5. IEEE.

[6] S. V. S. S. Lakshmi, S. D. Kavilla “Machine Learning For Credit Card Fraud Detection System”, unpublished

[7] N. Malini, Dr. M. Pushpa, “Analysis on Credit Card Fraud Identification Techniques based on KNN and Outlier Detection“, Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017 Third International Conference on pp. 255- 258. IEEE.

[8] Mrs. C. Navamani, M. Phil, S. Krishnan, “Credit Card Nearest Neighbor Based Outlier Detection Techniques”

[9] J. O. Awoyemi, A. O. Adentumbi, S. A. Oluwadare, “Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis”, Computing Networking and Informatics (ICCNI), 2017 International Conference on pp. 1-9. IEEE.

[10] Z. Kazemi, H. Zarrabi, “Using deep networks for fraud detection in the credit card transactions”, Knowledge-Based Engineering and Innovation (KBEI), 2017 IEEE 4th International Conference on pp. 630-633. IEEE.