CREDIT CARD FRAUD DETECTION

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

JOYJIT GUHA SIR

BY

BRISTIDEV BURMAN

MAFUJA KHATUN

GURU NANAK INSTITUTE OF TECHNOLOGY

In association with

Module 132 ,SDF Building Saltlake Sector V,Kolkata 700091

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1. Title of the Project: **CREDIT CARD FRAUD DETECTION**

2.Project Members: I. BRISTIDEV BURMAN
II. MAFUJA KHATUN

3. Name of the guide: **Mr. JOYJIT GUHA**

4.Address: Ardent Computech Pvt. Ltd(An ISO 9001:2015 Certified)
Module 132 ,SDF Building Saltlake Sector V,Kolkata 700091

Project Version Control History

Version	Members	Description Of Version	Date Completed
Final	1.BRISTIDEV BURMAN 2.MAFUJA KHATUN	Project Report	

Signature of Team Member

Date:

Signature of Approver

Date:

MR.JOYJIT GUHA

Project Proposal Evaluator

For Office Use Only

APPROVED	**NOTAPPROVED**

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled

"CREDIT CARD FRAUD DETECTION "

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

At

Module 132 ,SDF Building Saltlake Sector V,Kolkata 700091

is an authentic work carried out under the guidance of

MR. JOYJIT GUHA.

The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:

Name of the Student: **BRISTIDEV BURMAN**
MAFUJA KHATUN

Signature of the students:

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Katakana, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled

"CREDIT CARD FRAUD DETECTION"

is a record of bonafide work, carried out by

BRISTIDEV BURMAN

MAFUJA KHATUN

under my guidance at **ARDENT COMPUTECH PVT LTD**.

In my opinion, the report in its present form is in partial fulfilment ofthe requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®.**

To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. JOYJIT GUHA

Project Engineer

Ardent ComputechPvt. Ltd (An ISO 9001:2015 Certified)

Module 132 ,SDF Building Saltlake Sector V,Kolkata 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work. I would like to show our greatest appreciation to

MR.JOYJIT GUHA,
Project Engineer at Ardent Computech, Kolkata.
I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and
guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

ABOUT PYTHON

Python is a high-level, general-purpose programming language,designed by Guido van Rossum in the late 80s and early 90s .It is designed to be highly readable and easy to learn for beginners.Python is derived from many other languages, including ABC, Modula-3, C.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compileyour program before executing it.

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 programmersand supports the development of a wide range of applications from simple text processing to WWW browsers to games.

FEATURES

1. -**Easy-to-learn:** Python has few Keywords, simple structure and clearly defined syntax.

2. **Portable**: Python can run on the wide variety of hardware platforms and has the same interface on all platforms

3.It can be easily **integrated** with C, C++, COM, ActiveX, CORBA and JAVA.

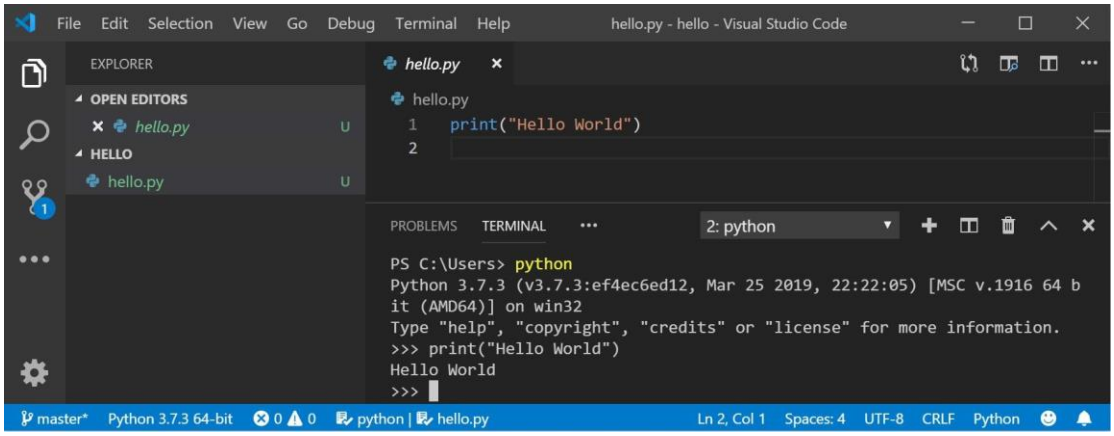
4.It can support functional and structured programming methods as well as Object Oriented Programming(OOP).

5. **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.

6. **Extendable**: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

7.It support functional and structured programming methods as well as OOP.

ENVIRONMENT SETUP



Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –
>>> print "Hello, Python!"
If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –
Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers.
Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use
them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only. **And, exec, not**
Assert, finally, or
Break, for, pass Class,
from, print continue,
global, raise def, if,
return del, import,
try elif, in, while else,
is, with except,
lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example – if True: print "True"
else:
print "False"

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −
$ python-h usage: python [option]...[-c cmd|-m mod | file |-][arg]...
Options and arguments (and corresponding environment variables):
-c cmd: program passed in as string(terminates option list)
-d : debug output from parser (also PYTHONDEBUG=x)
-E : ignore environment variables (such as PYTHONPATH)
-h : print this help message and exit [etc.]

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10
# An integer assignment
weight=10.60 # A
floating point
name="Ardent" # A
string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

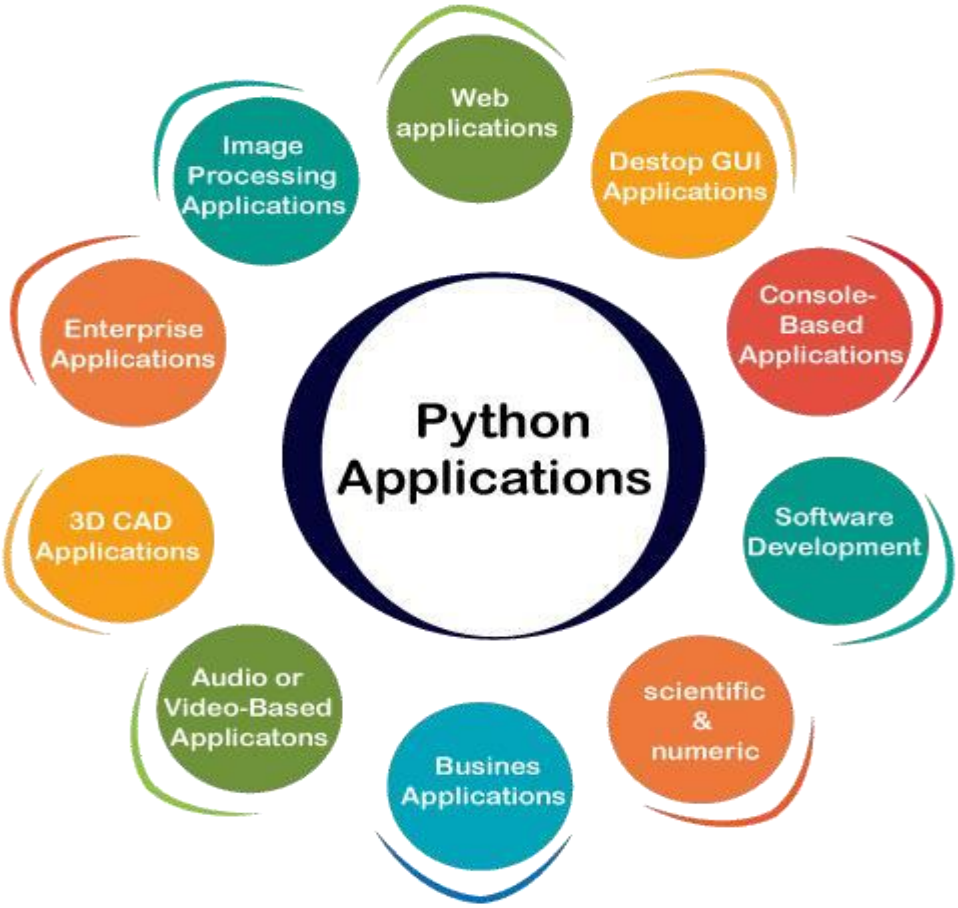
The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters.

Python has five standard data types −

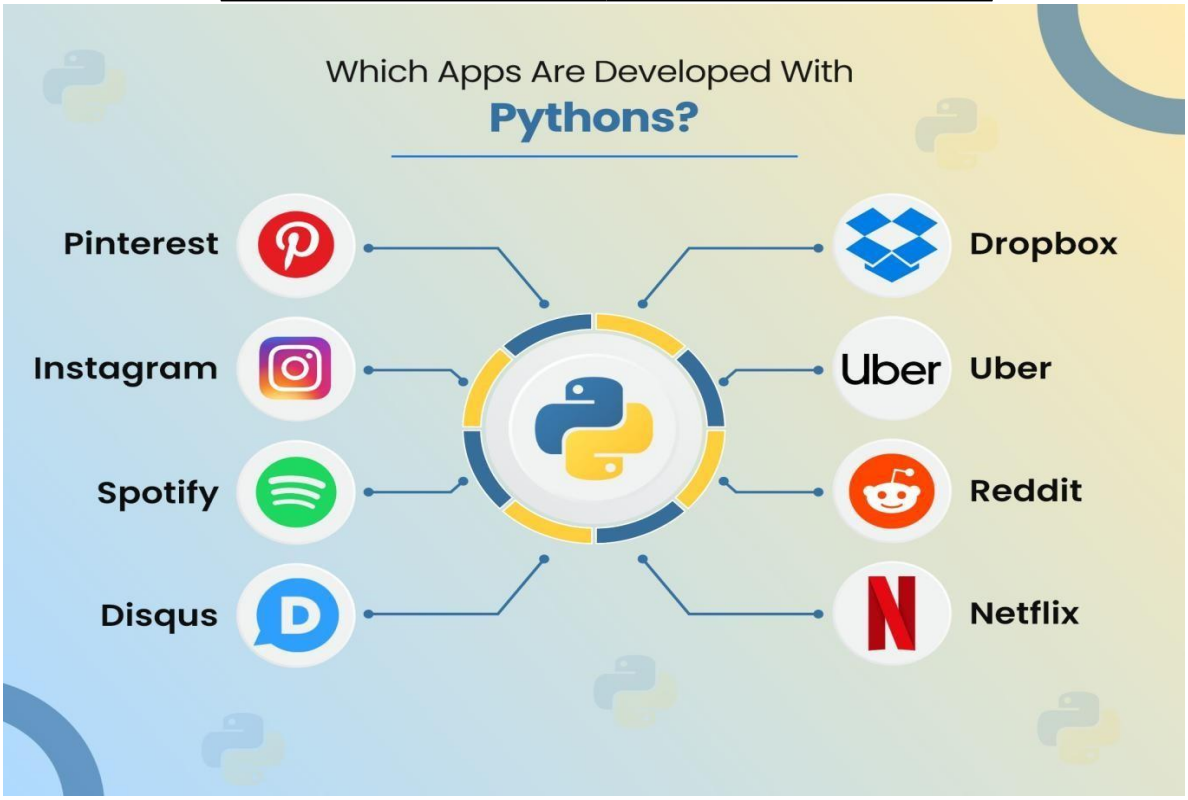
1.String
2.List
3.Tuple
4.Dictionary
5.Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function. There are several built-in functions to perform conversion from one one data type to another.

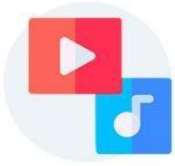


APPLICATIONS OF PYTHON



APPS THAT USE PYTHON

What Type of Apps Can You Develop in Python?



Audio-video apps

Game app development

Blockchain Application

Command-line apps

Machine learning apps

Business apps

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object.

An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9). Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use
them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only. **And, exec, not**
Assert, finally, or
Break, for, pass Class,
from, print continue,
global, raise def, if,
return del, import,
try elif, in, while else,
is, with except,
lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True: print "True" else:
print "False"
```

FUNCTIONS

Defining a Function

 deffunctionname(parameters):
"function_docstring"
function_suite return
[expression]

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

 # Function definition is here
defchangeme(mylist):
"This changes a passed list into this function"
mylist.append([1,2,3,4]); print"Values inside
the function: ",mylist return

 # Now you can call changeme function
mylist=[10,20,30]; changeme(mylist);
print"Values outside the function: ",mylist
Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result − Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example
total=0;

 # This is global variable.

 # Function definition is here
def sum(arg1, arg2):
*# **Add both the parameters and return them."***

total= arg1 + arg2;# Here total is local variable.
print"Inside the function local total : ", total
return total;

Now you can call sum function sum(10,20);
print"Outside the function global total : ", total
When the above code is executed, it produces the following result −
Inside the function local total : 30
Outside the function global total : 0

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .
The Python code for a module named *aname* normally resides in a file named *aname.py*.

Here's an example of a simple module, support.py
def print_func(par): print"Hello : ", par return

The *import* Statement
You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax – import module1[, module2[,… moduleN]

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on. Consider a file *Pots.py* available in *Phone* directory. This file has following line of source

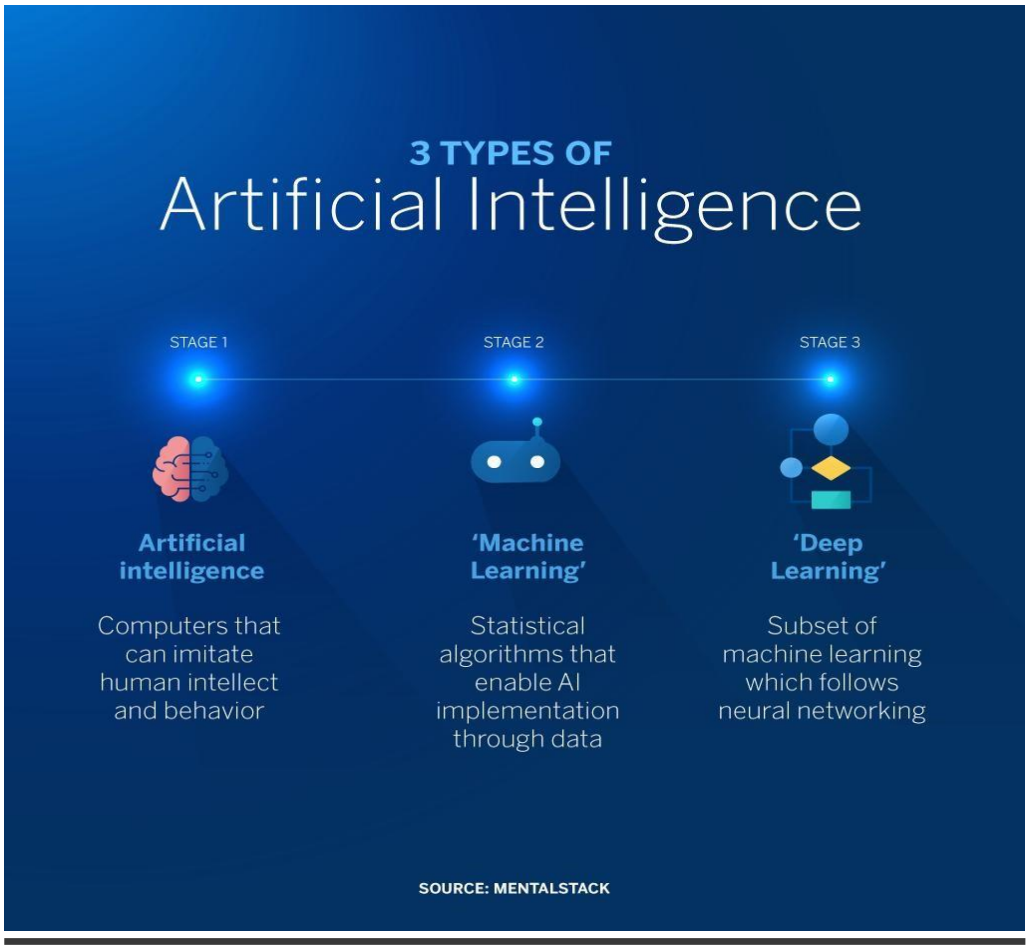
code − def
Pots():
print "I'm Pots Phone"
Similar way, we have another two files having different functions with the same name as above –
Phone/Isdn.py file having function Isdn()

Phone/G3.py file having function G3()

Now, create one more file __init__.py in *Phone* directory −
Phone/__init__.py
To make all of your functions available when you've imported Phone, you need to put
explicit import statements in __init__.py as follows −
from Pots import Pots from Isdn import Isdn from
G3 import

ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is *"The science and engineering of making intelligent machines, especially intelligent computer programs"*.

Artificial Intelligence is a way of making a computer, a computercontrolled robot, or a software think intelligently, in the similar manner the intelligent humans think.

Goals of AI

>To Create Expert Systems − The systems which exhibit intelligent behaviour, learn,demonstrate, explain, and advice its users.
>To Implement Human Intelligence in Machines − Creating systems that understand, think,learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as :-

Gaming − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing − It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems − These systems understand, interpret, and comprehend visual input on the computer.
For example:
1.A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas.
2.Doctors use clinical expert system to diagnose the patient.
3.Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

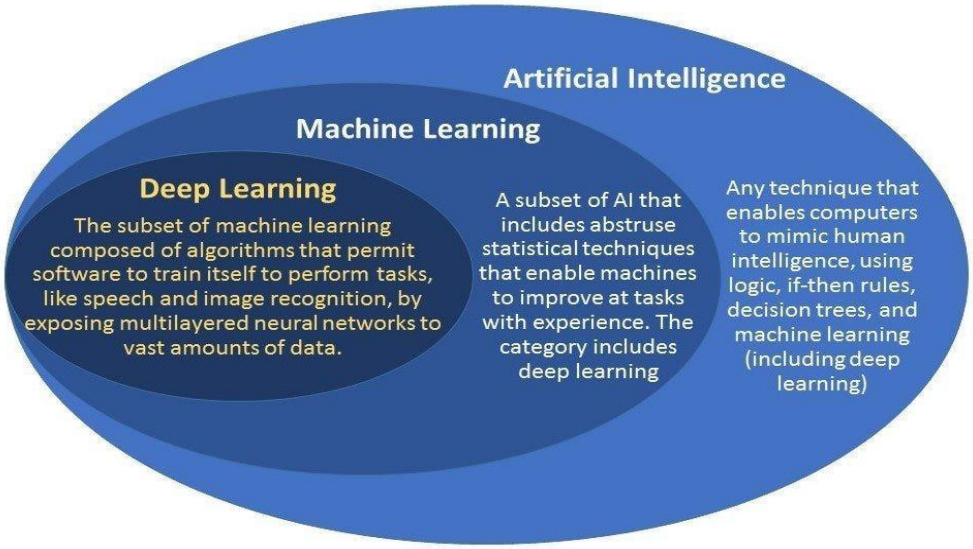
Speech Recognition − Some intelligent systems are capable of hearing and comprehending
the language in terms of sentences and their meanings while a human talks to it. It can handle
different accents, slang words, noise in the background, change in human's noise due to cold, etc.
Handwriting Recognition − The handwriting recognition software reads the text written on

paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

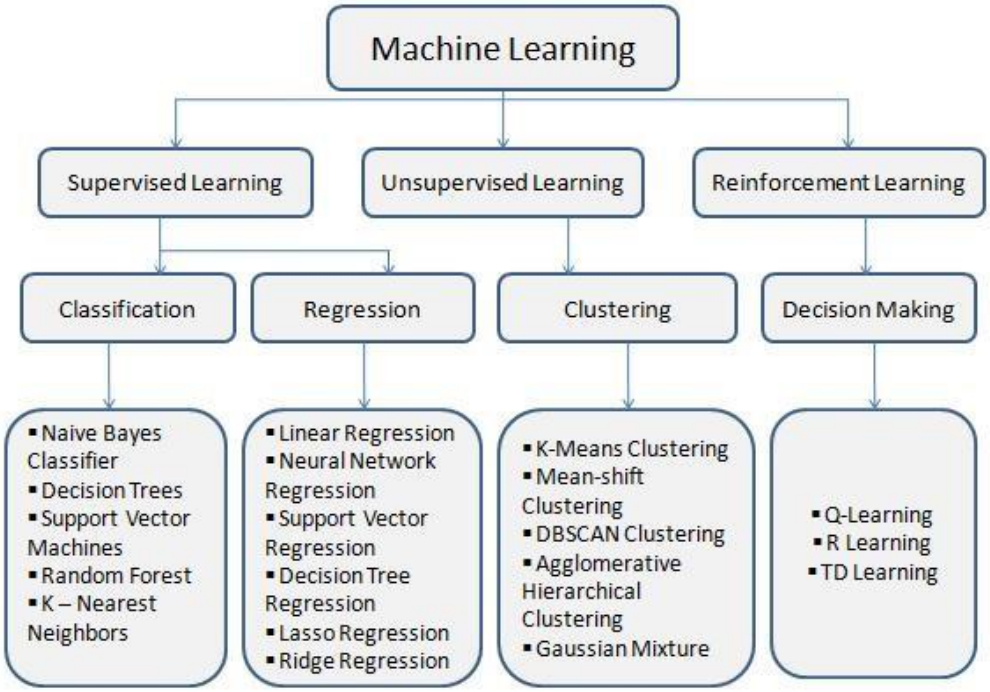
Intelligent Robots − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment. **Deep Learning**

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning. Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- use some form of gradient descent for training via backpropagation.



MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

.

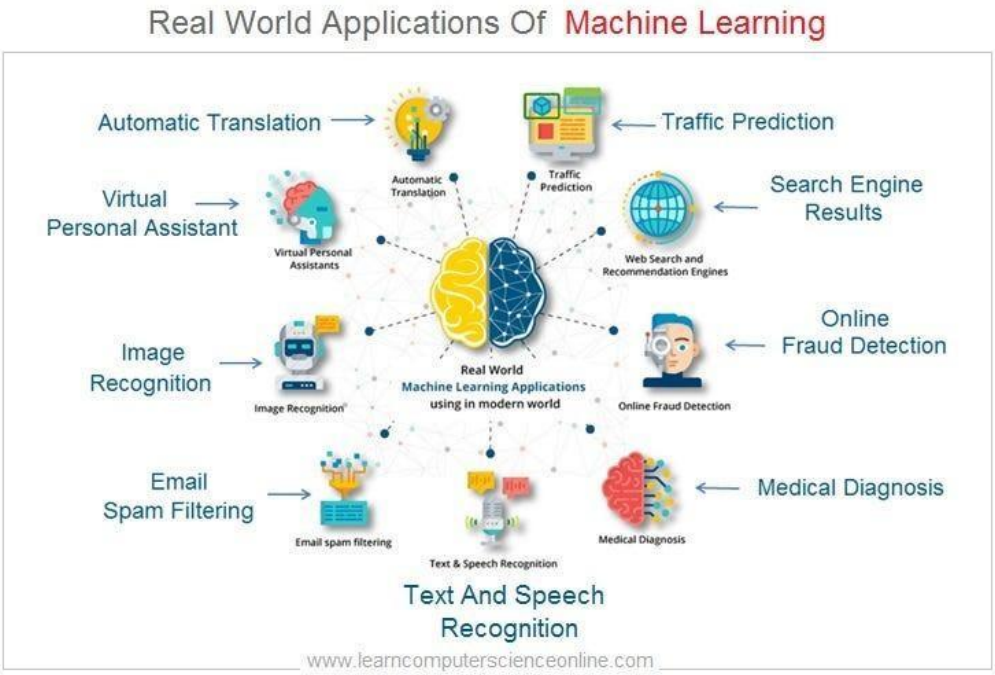
SUPERVISED LEARNING

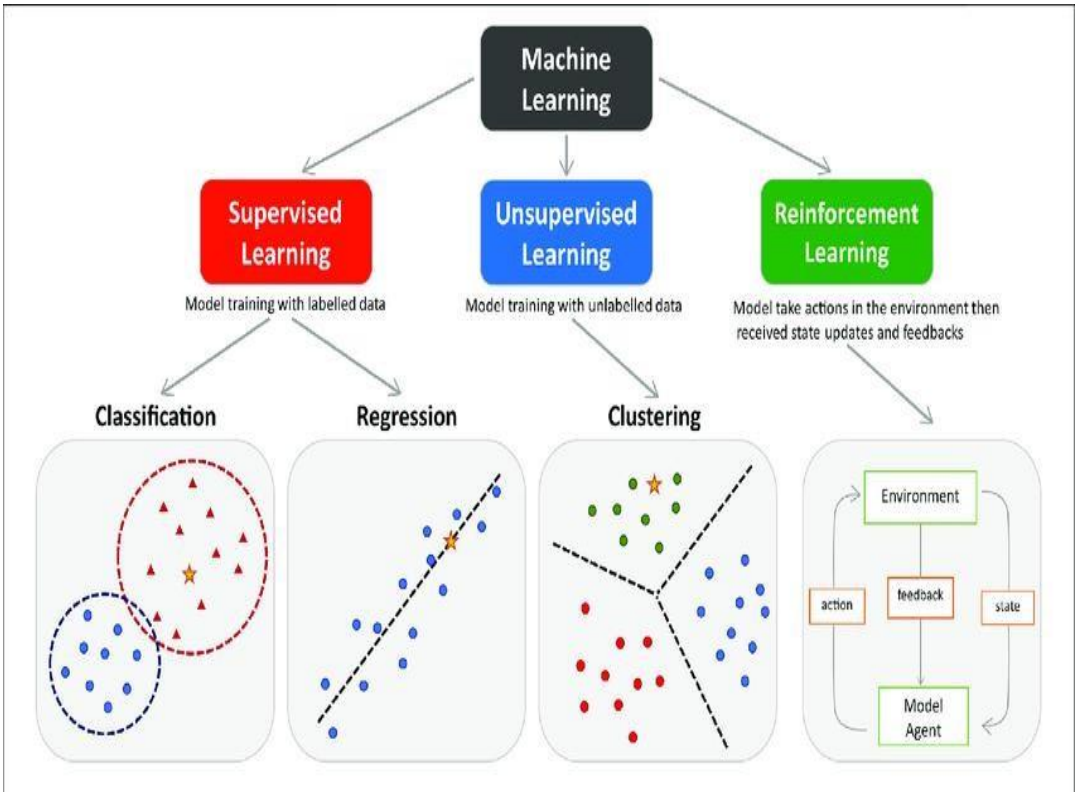
Supervised learning is the machine learning task of inferring a function from *labeled training data*. The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

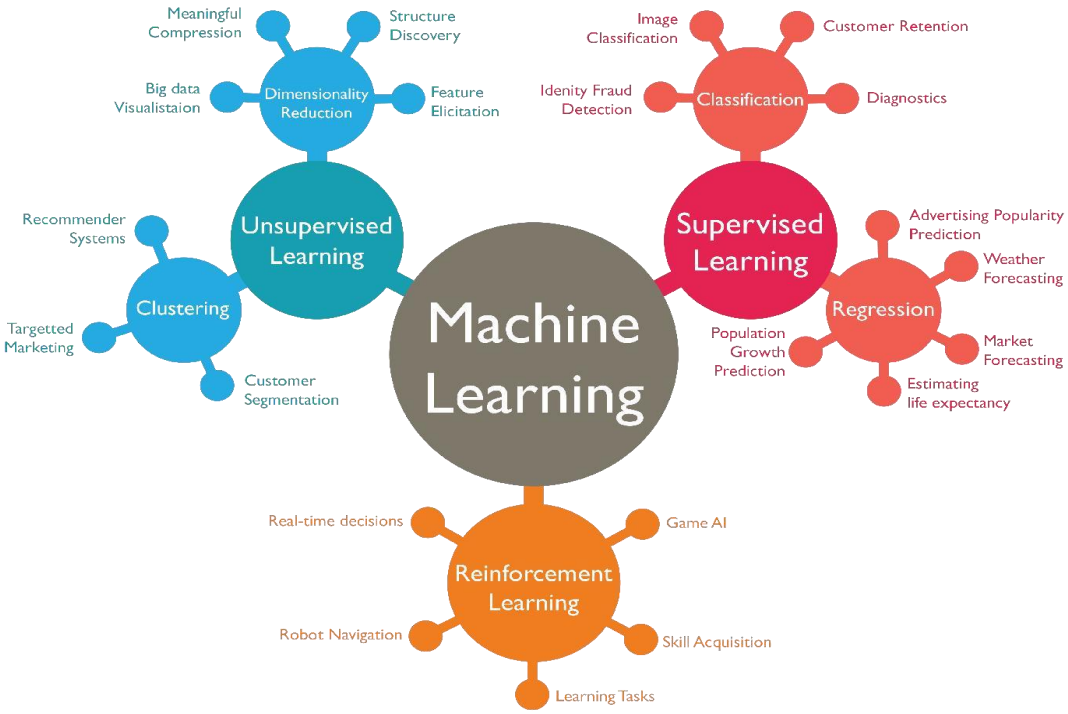
A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning. A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.




Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Model training with labelled data
Model training with unlabelled data
Model take actions in the environment then received state updates and feedbacks
Classification
Regression
Clustering
Environment
action
feedback
state
Model Agent


Machine Learning
Unsupervised Learning
Meaningful Compression
Structure Discovery
Dimensionality Reduction
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Image Classification
Customer Retention
Classification
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Estimating life expectancy
Population Growth Prediction
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

NUMPY

NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2- dimensional).
The first dimension (axis) has a length of 2, the second dimension has a length of 3. [[1., 0., 0.], [0., 1., 2.]]
NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

```
importnumpy as np
a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print 'Our array is:' print a
print '\n'
print 'The items in the second column are:'
print a[...,1]
print '\n'
print 'The items in the second row are:'
print a[1,...]
print '\n'
print 'The items column 1 onwards are:'
print a[...,1:]
```

OUTPUT

```
Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]
The items in the second column are:
[2 4 5]
The items in the second row are:
[3 4 5]
The items column 1 onwards are:
[[2 3] [4 5]
[5 6]]
```

SCIPY

>Modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

>SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries.

>This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

.Data Structures

>The basic data structure used by SciPy is a multidimensional array provided by the NumPy module.
>NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy.
>NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

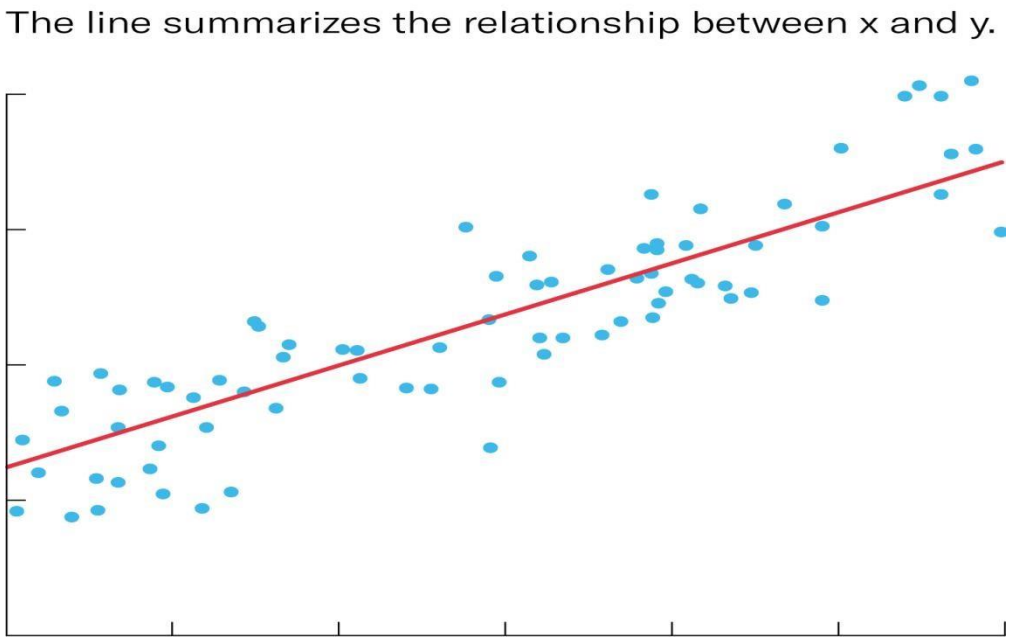
SCIKIT-LEARN

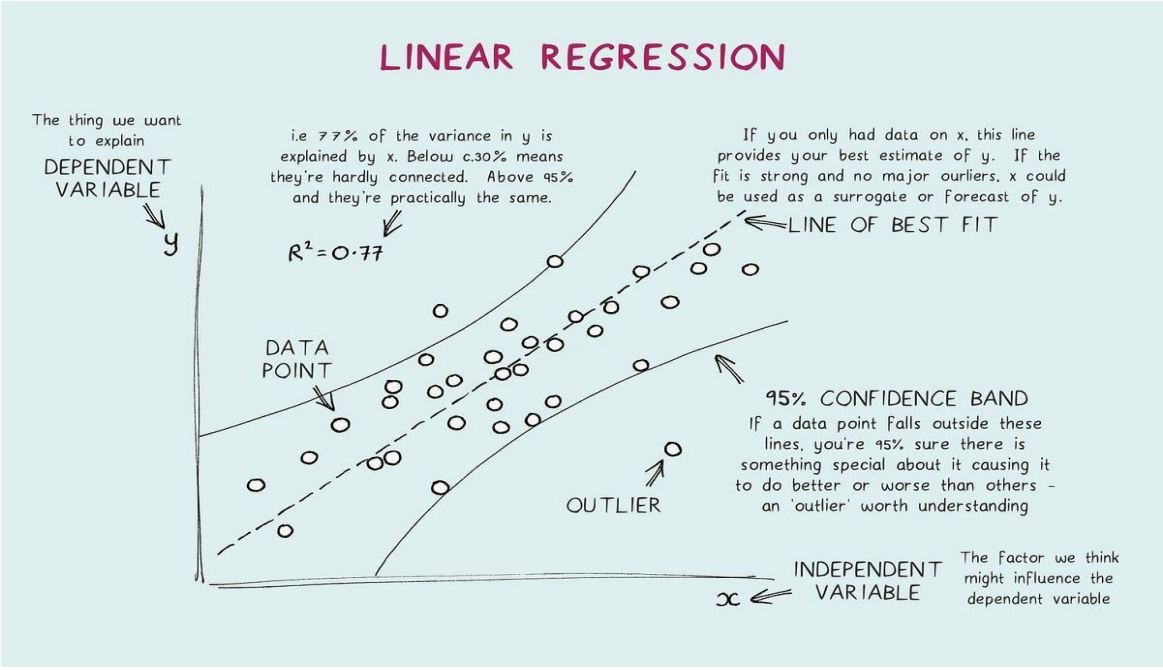
Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

REGRESSION ANALYSIS

In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed. Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances,

regression analysis can be used to infer casual relationships between the independent and dependent variables.





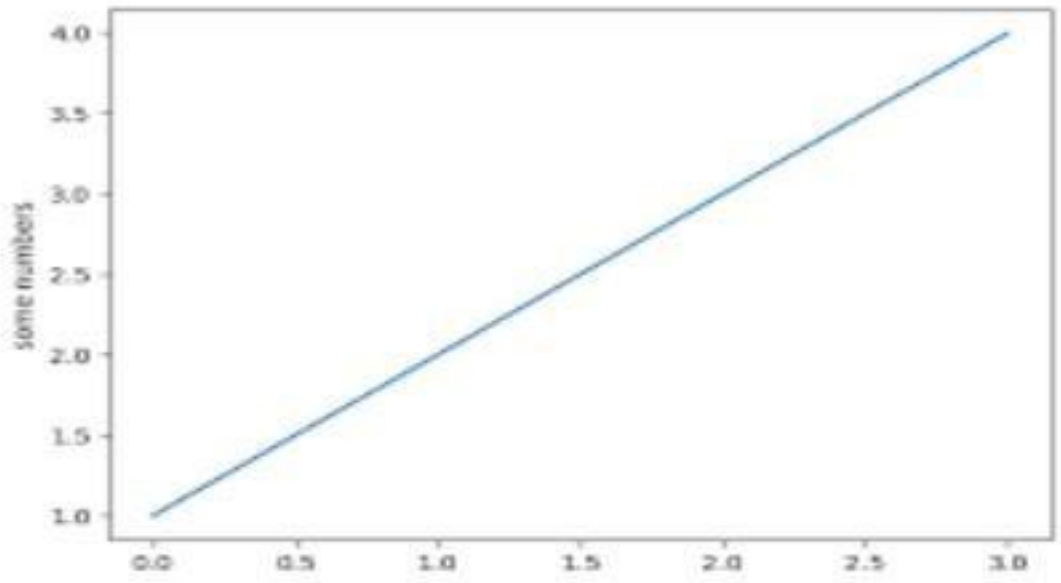
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an objectoriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

EXAMPLE

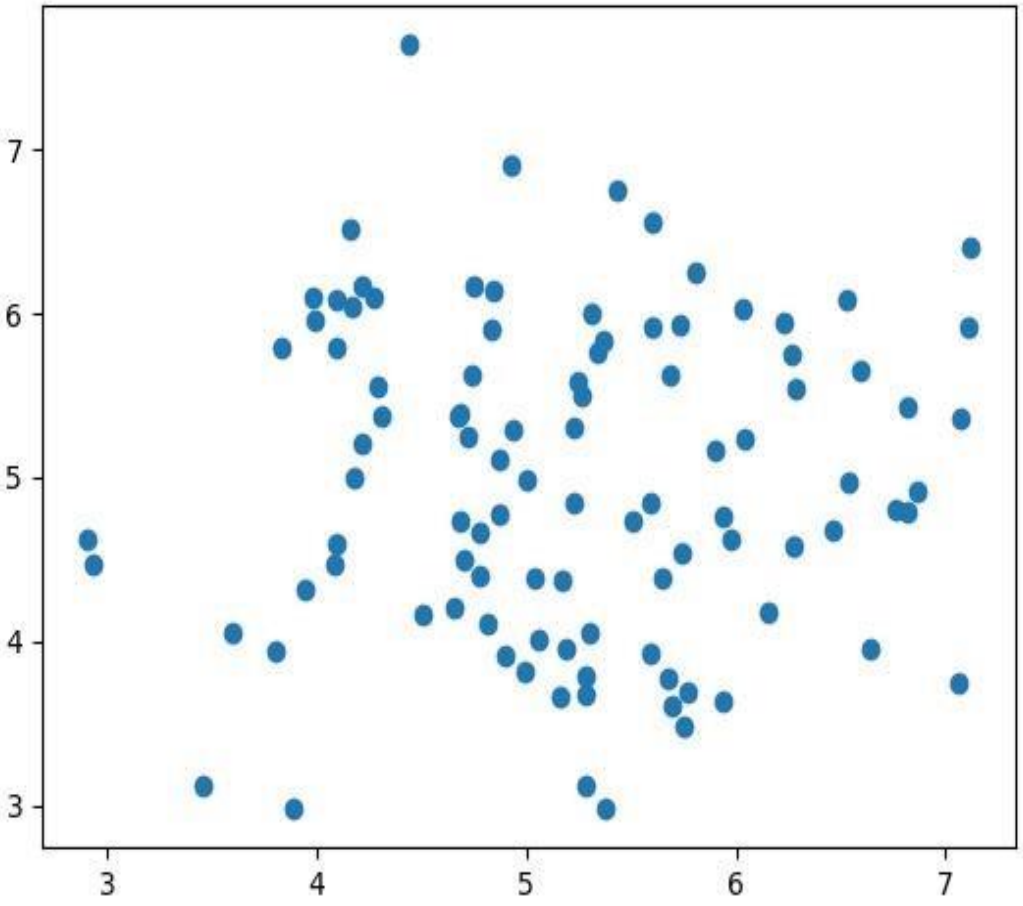
- **LINE PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



➢ SCATTER PLOT

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



LOGISTIC REGRESSION

>.Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

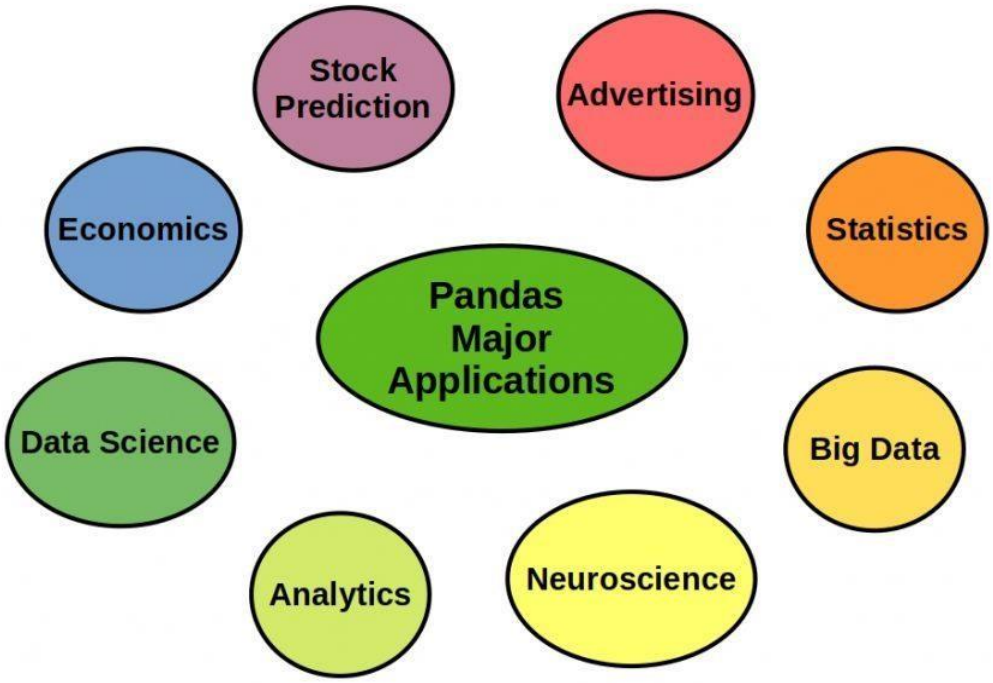
>.Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x.

>.Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

>.Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional,structured data sets.



LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion. ➢ Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

>.Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

>.Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions.

>.Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from an online website,NOAA(National Oceanic And Atmospheric Adminstration.
We have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Lasso Linear Model and Bayesian Ridge Linear Model.

Training: The data sets was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict . Both y_train and y_predict was compared.

CREDIT CARD FRAUD DETECTION

1. **Credit card fraud detection** is the process of identifying unauthorized or fraudulent activity on credit card accounts through advanced algorithms and machine learning techniques. It involves monitoring transactions, detecting unusual patterns, and implementing security measures to prevent financial losses for both cardholders and issuers.

2. **Credit card fraud** involves the unauthorized use of someone else's credit card information for financial gain, typically through deceptive or illegal means such as stolen cards, counterfeit cards, or identity theft. It can result in financial losses for cardholders, merchants, and financial institutions, as well as potential damage to credit scores and reputations. Detection and prevention efforts are crucial to mitigate these risks.

3. **Credit card fraud detection** primarily utilizes data analytics, machine learning algorithms, and pattern recognition to identify suspicious transactions and behaviors, aiming to minimize fraudulent activity and protect consumers and financial institutions.

4. Some of its use cases include-

- Online purchases: Analyzing online transaction patterns.
- Point-of-sale transactions: Real-time monitoring for suspicious behavior.
- ATM withdrawals: Detecting unauthorized cash withdrawals.
- Travel bookings: Verifying legitimacy of travel transactions.

MODEL TRAINING

1. **Data collection**: Gather a diverse dataset containing both legitimate and fraudulent credit card transactions.
2. **Feature engineering**: Extract relevant features such as transaction amount, location, time, and user behavior.
3. **Model selection**: Choose appropriate machine learning algorithms like logistic regression, random forests, or neural networks.
4. **Training**: Train the selected models on the labeled dataset, optimizing hyperparameters for performance.
5. **Evaluation**: Assess model performance using metrics like accuracy, precision, recall, and F1-score.
6. **Validation**: Validate models on separate datasets to ensure generalization and robustness.
7. **Deployment**: Deploy the trained model into production systems for real-time fraud detection.

ACTUAL CODE FOR CREDIT CARD FRAUD DETECTION

1.USING DECISION TREE ALGORITHM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
credit=pd.read_csv('/content/drive/MyDrive/DATASETS/creditcard.csv')
credit
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0.77	0
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	24.79	0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	67.88	0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	10.00	0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.013649	217.00	0

84807 rows × 31 columns

```
[ ] X=credit.drop(['Class','Amount','Time'], axis=1)
    Y=credit['Class']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2)
print('Training Data',X_train)
print('Testing Data',X_test)
print('Training Data y',y_train)
print('Testing Data y',y_test)
```

```
Training Data              V1        V2        V3        V4        V5        V6        V7  \
218375  0.095825  0.869604  0.011025 -0.750683  0.767399 -0.726437  1.009362
69774   -0.604237  1.225626  0.882227  0.946374 -0.310060 -0.743293  0.610900
55891    0.319748 -1.544982  0.831751  1.005981 -0.845023  1.695313 -0.537292
255266 -1.337355 -1.005519 -2.571016 -1.885885  1.244514 -2.744724  1.607383
205468 -1.257380  0.148427  1.122478 -3.364173 -0.605994 -0.621323 -0.105895
...          ...       ...       ...       ...       ...       ...       ...
202900 -0.491873  1.356494 -0.751957  0.651015  1.242345  0.153067  0.777299
195505 -0.601666  0.360018 -0.047917 -0.635793  0.947430 -0.715207  0.888914
174159  1.929025  0.626157 -0.970011  3.604050  0.771009  0.036124  0.317972
155751  1.873928 -0.432011 -0.511812  0.132338  0.212997  1.308348 -0.961770
187144  1.867120 -0.780505 -0.783968  0.051674 -0.433185  0.094086 -0.626569

                V8        V9       V10  ...       V19       V20       V21  \
218375 -0.176338 -0.032221 -0.415651  ...  0.175286 -0.004419 -0.304127
69774   0.242554 -0.848246 -0.482374  ...  0.418167 -0.022113  0.224692
55891   0.620558  1.114295 -0.639974  ... -1.549280  0.371815  0.123318
255266 -0.162163  0.126063 -1.150747  ... -1.106866  0.384825  0.726059
205468  0.592577 -0.718946 -0.765292  ... -1.213094 -0.562012 -0.346840
...          ...       ...       ...  ...       ...       ...       ...
202900  0.468954 -0.939049 -0.028822  ...  1.300175 -0.033781  0.153629
195505 -0.053178  0.252844 -0.298003  ...  0.063472 -0.141560 -0.184571
174159 -0.140310 -1.214005  1.513338  ... -1.787907 -0.207447  0.238264
155751  0.485894  2.552909 -0.526994  ... -0.882352 -0.422963 -0.190687
187144  0.106502  0.955951  0.208004  ...  0.244994 -0.002245  0.212862
```

```
            V22       V23       V24       V25       V26       V27       V28
218375 -0.679169 -0.021956 -0.655017 -0.389649  0.180397  0.248317  0.090801
69774   0.496175 -0.011900  0.391341 -0.215631 -0.333734  0.033594  0.105074
55891  -0.042301 -0.062550 -0.603309 -0.256598  0.450391  0.011757  0.068786
255266  1.386385  0.428832  0.065108 -1.011275  0.530357  0.267713  0.102775
205468 -0.679966 -0.315578 -0.369434  0.597762  0.560043  0.067908  0.004793
...          ...       ...       ...       ...       ...       ...       ...
202900  0.525056 -0.306434  0.029006  0.095787 -0.382420  0.305326  0.154866
195505 -0.388583  0.381539  0.567510 -0.799077  0.133970  0.278123  0.268849
174159  0.657963  0.079720  0.674716  0.148192  0.101777 -0.047553 -0.042692
155751 -0.143505  0.371802 -1.782084 -0.588194 -0.846958  0.070476 -0.062936
187144  0.503694 -0.006717 -0.924714 -0.305487  0.683774 -0.060668 -0.054308

[227845 rows x 28 columns]
Testing Data               V1        V2        V3        V4        V5        V6        V7  \
176539 -0.175144  1.553927 -1.504921 -0.324668  2.081521 -1.539445  1.725862
281486  2.098942 -0.879323 -1.134417 -2.332672 -0.482294 -0.303863 -0.777325
197228  1.987640 -0.501336 -0.234145  0.500429 -0.905020 -0.588915 -0.695366
217392 -0.740571  1.511599 -0.797272 -0.423118 -0.005841 -0.989269  0.265900
79739  -0.337043  0.503585  1.097722 -1.369642 -0.025873 -0.673068  0.377161
...          ...       ...       ...       ...       ...       ...       ...
66779   1.029946  0.039886  0.471571  1.358789 -0.226768  0.036229  0.028452
182605 -1.515794  1.732805  0.689584  2.679154  2.017643  0.130093  1.624213
200782  1.985970 -0.463859 -0.540295  0.385604 -0.472396 -0.204198 -0.563511
146424 -5.210295  4.916165 -3.063086 -3.137821  1.422862 -0.653995  2.342638
11509   1.386986 -0.421137  0.551988 -0.452969 -0.737580 -0.245732 -0.703880

             V8        V9       V10  ...       V19       V20       V21  \
176539 -0.421707 -0.888006 -1.819798  ... -0.387906  0.000323  0.048660
281486  0.087593  2.453985 -1.082120  ...  1.261498 -0.189067  0.233404
197228 -0.019891  1.369771  0.026956  ... -0.241330 -0.229396  0.150991
217392  0.649148 -0.632011 -0.471529  ...  0.214308 -0.223301  0.540618
79739   0.122249  0.459084 -0.602459  ... -0.432347 -0.073484 -0.179734
...          ...       ...       ...  ...       ...       ...       ...
66779   0.102825  0.002295  0.023316  ... -0.168603 -0.100337 -0.011516
182605 -0.260374 -2.782087  1.348879  ... -0.634460 -0.356731  0.115685
200782 -0.053803  1.454676 -0.182372  ... -0.012023 -0.129426  0.221239
146424 -1.089480  5.516694  9.008699  ... -0.263387  3.635146 -0.875794
11509  -0.127466  0.481259  0.268122  ...  1.146659  0.066494 -0.135536
```

```
            V22       V23       V24       V25       V26       V27       V28
176539  0.264088 -0.377570  0.577560  0.053781  0.599165  0.098910  0.225387
281486  0.885803  0.035565  0.105877  0.090857 -0.755611  0.065585 -0.048510
197228  0.625925  0.163502  0.016210 -0.296404  0.572293 -0.023802 -0.048383
217392  1.515020 -0.019226  0.070319 -0.792367 -0.249507  0.127617  0.182293
79739  -0.576470 -0.090461 -0.469035 -0.206015  0.781259  0.159876  0.055164
...          ...       ...       ...       ...       ...       ...       ...
66779   0.079219 -0.085973  0.236152  0.593140 -0.346799  0.029299  0.014927
182605  0.334037 -0.210247  0.775352  1.071887  0.195755 -0.460151 -0.014991
200782  0.888780 -0.026047 -0.606215  0.053212 -0.117098  0.040001 -0.039894
146424  0.252991 -0.175645 -0.466692  0.933246  0.061088  2.140386  0.469747
11509  -0.032976 -0.080614 -0.014218  0.578013 -0.281732 -0.010756 -0.005852

[56962 rows x 28 columns]
Training Data y 218375    0
69774     0
55891     0
255266    0
205468    0
         ..
202900    0
195505    0
174159    0
155751    0
187144    0
Name: Class, Length: 227845, dtype: int64
Testing Data y 176539    0
281486    0
197228    0
217392    0
79739     0
         ..
66779     0
182605    0
200782    0
146424    0
11509     0
Name: Class, Length: 56962, dtype: int64
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
    classifier=DecisionTreeClassifier()
    classifier.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[ ] y_pred=classifier.predict(X_test)
    y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred, target_names=['FAKE','AUTHENTIC']))
```

```
[[56864    23]
 [   22    53]]
              precision    recall  f1-score   support

        FAKE       1.00      1.00      1.00     56887
   AUTHENTIC       0.70      0.71      0.70        75

    accuracy                           1.00     56962
   macro avg       0.85      0.85      0.85     56962
weighted avg       1.00      1.00      1.00     56962
```

```
[ ] y_pred=classifier.predict([[-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.338320769942518,0.462387777762292,0.239598554061257,0.0986979012610507,0.363786969611213,
                                0.0907941719789316,-0.551599533260813,-0.617800855762348,-0.991389847235408,-0.311169353699879,1.46817697209427,-0.470400525259478,0.207971241929242,0.0257905801985591,
                                0.403992960255733,0.251412098239705,-0.018306777944153,0.277837575558899,-0.110473910188767,0.0669280749146731,0.128539358273528,-0.189114843888824,0.133558376740387,
                                -0.0210530534538215]])
    print(y_pred)
```

```
[0]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

```
[ ] import pandas as pd
    df1=pd.read_csv('/content/drive/MyDrive/DATASETS/CC.csv', index_col=0)
    df1.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
1	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
2	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
3	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
4	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
5	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

```
df1=df1.drop(['Time','Amount','Class'],axis=1)
df1.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28
1	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	0.403993	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053
2	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.145783	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724
3	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	-2.261857	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752
4	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-1.232622	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458
5	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	0.803487	0.408542	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153

5 rows × 28 columns

```
df1=df1.to_numpy()
df1
```

```
array([[-1.35980713e+00, -7.27811733e-02,  2.53634674e+00, ...,
        -1.89114844e-01,  1.33558377e-01, -2.10530535e-02],
       [ 1.19185711e+00,  2.66150712e-01,  1.66480113e-01, ...,
         1.25894532e-01, -8.98309914e-03,  1.47241692e-02],
       [-1.35835406e+00, -1.34016307e+00,  1.77320934e+00, ...,
         -1.39096572e-01, -5.53527940e-02, -5.97518406e-02],
       ...,
       [ 1.91956501e+00, -3.01253846e-01, -3.24963981e+00, ...,
         -8.73705959e-02,  4.45477214e-03, -2.65608286e-02],
       [-2.40440050e-01,  5.30482513e-01,  7.02510230e-01, ...,
         5.46668462e-01,  1.08820735e-01,  1.04532821e-01],
       [-5.33412522e-01, -1.89733337e-01,  7.03337367e-01, ...,
         -8.18267121e-01, -2.41530880e-03,  1.36489143e-02]])
```

```
y_pred=classifier.predict(df1)
print(y_pred)
```

```
[0 0 0 ... 0 0 0]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

FINDS AUTHENTIC CREDIT CARDS

```
import numpy as np
np.where(y_pred == 1)
```

```
(array([   541,    623,   4920,   5171,   6108,   6329,   6331,   6334,
          6336,   6338,   6427,   6446,   6472,   6529,   6609,   6641,
          6717,   6719,   6734,   6774,   6820,   6870,   6882,   6899,
          6903,   6971,   8296,   8312,   8335,   8615,   8617,   8842,
          8845,   8972,   9035,   9179,   9252,   9487,   9509,  10204,
         10484,  10497,  10498,  10568,  10630,  10690,  10801,  10891,
         10897,  11343,  11710,  11841,  11880,  12070,  12108,  12261,
         12369,  14170,  14197,  14211,  15166,  15204,  15225,  15321,
         15451,  15476,  15506,  15539,  15566,  15736,  15751,  15781,
         15810,  16415,  16780,  16863,  17317,  17366,  17407,  17453,
         17480,  18466,  18472,  18773,  18809,  20198,  23308,  23422,
         24520,  27359,  27362,  27627,  27738,  27749,  29687,  30100,
         30314,  30384,  30398,  30442,  30473,  30496,  31002,  33276,
         40085,  40525,  41395,  41569,  41943,  42007,  42009,  42473,
         42528,  42549,  42590,  42609,  42635,  42674,  42696,  42700,
         42741,  42756,  42769,  42784,  42856,  42887,  42936,  42945,
         42958,  43061,  43160,  43204,  43428,  43624,  43681,  43773,
         44001,  44223,  44270,  44556,  45203,  45732,  46578,  46909,
         46918,  46998,  47802,  48094,  50211,  50509,  50537,  52466,
         52584,  53591,  53794,  55401,  56703,  57248,  57470,  57615,
         58422,  58761,  59539,  61787,  63421,  63634,  64329,  64411,
         64460,  68320,  68522,  68633,  69498,  69980,  70141,  70589,
         72757,  73784,  73857,  74496,  74507,  74794,  75511,  76555,
```

2.USING LOGISTIC REGRESSION ALGORITHM

```
[ ] import numpy as np
    import pandas as pd
    credit_df=pd.read_csv('/content/drive/MyDrive/DATASETS/creditcard.csv')
```

```
[ ] credit_df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

```
[ ] X=credit_df.drop(['Time','Amount','Class'],axis=1)
    Y=credit_df['Class']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2)
print('Training Data',X_train)
print('Testing Data',X_test)
print('Training Data y',y_train)
print('Testing Data y',y_test)
```

```
Training Data             V1        V2        V3        V4        V5        V6        V7  \
252956 -3.692348  3.008841 -1.121494 -0.778320 -2.538427 -0.310044 -1.727557
194309 -0.964199  1.136658  0.759537 -0.800651  0.834249 -0.778122  1.105079
270255 -0.966559  1.591731  1.354527  4.026130  0.678443  3.500128 -0.608266
223988  0.066637  0.059921  0.374792 -1.822184  0.119898 -0.979894 -1.120815
138234 -1.791995  1.102738  0.324217  1.082267 -0.303348 -1.050303  0.066270
```

```
[ ] from sklearn.linear_model import LogisticRegression
    logreg=LogisticRegression()
    logreg.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
[ ] y_predict=logreg.predict(X_test)
    y_predict
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
pd.DataFrame({'Actual':y_test,'Predicted':y_predict})
```

	Actual	Predicted
250435	0	0
88337	0	0
191209	0	0
179001	0	0
253768	0	0
...	...	...
211773	0	0
104080	0	0
222355	0	0
149266	0	0
117057	0	0

```
[ ] from sklearn.metrics import classification_report, confusion_matrix
    print(confusion_matrix(y_test,y_predict))
    print(classification_report(y_test,y_predict, target_names=['FAKE','AUTHENTIC']))
```

```
[[56864     8]
 [   31    59]]
              precision    recall  f1-score   support

        FAKE       1.00      1.00      1.00     56872
   AUTHENTIC       0.88      0.66      0.75        90

    accuracy                           1.00     56962
   macro avg       0.94      0.83      0.88     56962
weighted avg       1.00      1.00      1.00     56962
```

```
df1=pd.read_csv('/content/drive/MyDrive/DATASETS/CC.csv', index_col=0)
df1=df1.drop(['Time','Amount','Class'],axis=1)
print(df1)
y_pred=classifier.predict(df1)
```

```
               V1        V2        V3        V4        V5        V6  \
1       -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388
2        1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361
3       -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499
4       -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203
5       -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921
...           ...       ...       ...       ...       ...       ...
284803 -11.881118 10.071785 -9.834783 -2.066656 -5.364473 -2.606837
284804  -0.732789 -0.055080  2.035030 -0.738589  0.868229  1.058415
284805   1.919565 -0.301254 -3.249640 -0.557828  2.630515  3.031260
284806  -0.240440  0.530483  0.702510  0.689799 -0.377961  0.623708
284807  -0.533413 -0.189733  0.703337 -0.506271 -0.012546 -0.649617

               V7        V8        V9       V10  ...       V19       V20  \
1        0.239599  0.098698  0.363787  0.090794  ...  0.403993  0.251412
2       -0.078803  0.085102 -0.255425 -0.166974  ... -0.145783 -0.069083
3        0.791461  0.247676 -1.514654  0.207643  ... -2.261857  0.524980
```

```
[ ] df_list=[]
    for i in y_pred:
      if i==0:
        temp_df = pd.DataFrame({'Authenticity': ['Fake']})
      else:
        temp_df = pd.DataFrame({'Authenticity': ['Real']})
      df_list.append(temp_df)
    df1=pd.concat(df_list,ignore_index=True)
```

```
df1
```

	Authenticity
0	Fake
1	Fake
2	Fake
3	Fake
4	Fake
...	...
284802	Fake
284803	Fake
284804	Fake
284805	Fake
284806	Fake

284807 rows × 1 columns

FINDS AUTHENTIC CREDIT CARDS

```
df1[df1['Authenticity']=='Real']
```

	Authenticity
541	Real
623	Real
4920	Real
5171	Real
6108	Real
...	...
279863	Real
280143	Real
280149	Real
281144	Real
281674	Real

493 rows × 1 columns

CONCLUSION

Credit card fraud detection employs advanced algorithms and machine learning techniques to safeguard against unauthorized transactions, mitigating financial losses and protecting consumers and financial institutions from fraudulent activity. Continuous innovation and vigilance are essential to stay ahead of evolving fraud tactics and ensure the security of credit card transactions.

This project verifies credit card authenticity and holds potential for various future enhancements, including the automatic validation of credit card authenticity during transactions on any website related to credit cards.