

**HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT**

**HAMIDPUR , DELHI - 110036**

**Affiliated to**

**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**

**Sector - 16C Dwarka, Delhi - 110075, India 2008-12**

**Summer Training Project Report**

**On**

**CREDIT CARD DETECTOR USING MACHINE LEARNING**

Submitted in partial Fulfilment of the requirements for

The award of the degree of

**Bachelor of Technology**

**In**

**(ELECRONICS AND COMMUNICATION ENGINEERING)**

**Guide(s): Submitted by:**

**LAKSHAY KUMAR Mr.Arpit Sodhi**

**00296502816**

**MANSI NARULA**

**41596502816**

**ANSHUL GUPTA**

**41396502816**

**SURYA SHARMA**

**43096502816**

DECLARATION

I/We, student(s) of B.Tech (ECE 4TH semester) hereby declare that the minor project entitled “Electricity Power Optimization And Bill Prediction” which is submitted to Department of CSE, HMR Institute of Technology & Management, Hamidpur Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka(New Delhi) in partial fulfilment of requirement for the award of the degree of Bachelor of Technology in ECE, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The list of member(s) involved in the project is listed below: -

1. MANSI NARULA (41596502816)
2. ANSHUL GUPTA(41396502816)
3. SURYA SHARMA(43096502816)
4. LAKSHAY KUMAR(00296502816)

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

|  |  |
| --- | --- |
| New Delhi | Mr. Trainer Name, Mr. Trainer Name (optional) |
| Date: | Signature of Guides |
|  |  |

**Mr. Ravinder Kumar**

**(Head of Department)**

**(Assistant Professor of Computer Science Department)**

**HMRITM Hamidpur, New Delhi-110036**

**Chapter – 1**

Team Details

LAKSHAY KUMAR

College Roll No. : 00296502816

Project Contribution:

"Contributed in finding the dataset for credit card's information. Designed the working algorithm and code along with Mansi. He visualized the data and helped in modifying the dataset as per the need of the algorithm, sorted out errors and helped partially in editing the report.

SURYA SHARMA

College Roll No. :43096502816

Project Contribution:

"He played a major role in editing the dataset. Adding some important

Components to the excel sheet. Made and helped Anshul with the User Interface

Of the project. Finally made the whole report for the project."

ANSHUL GUPTA

College Roll No. :41396502816

Project Contribution:

"He played a major role in editing the dataset. He helped with the coding part by utilising his error solving skills. Made the interface and helped Surya with the User Interface editing

Of the project. Helped with the correction of errors in the project. Mapped the code with the interface.”

MANSI NARULA

College Roll No. :41596502816

Project Contribution:

"She contributed in finding the appropriate dataset from different sources. She was a part of

algorithm designing and code content along with Lakshay and also helped in editing some

modifications of dataset as required by the algorithm and also in overall management of

project."

**Chapter – 2**

Abstract

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.

In our project ,we have attempted to illustrate the modeling of a data set using a machine learning classification, with Credit Card Fraud Detection being the base. Classification is a machine learning paradigm that involves deriving a function that will separate data into categories, or classes, characterized by a training set of data containing observations (instances) whose category membership is known. This function is then used in identifying in which of the categories a new observation belongs.

Credit Card Fraud Detection is a typical example of classification. In this process, we have focused more on analyzing the feature modeling and possible business use cases of the algorithm’s output than on the algorithm itself. We used the implementation of Logistic Regression Algorithm.

The Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

**Chapter – 3**

Overview and Algorithms Used:

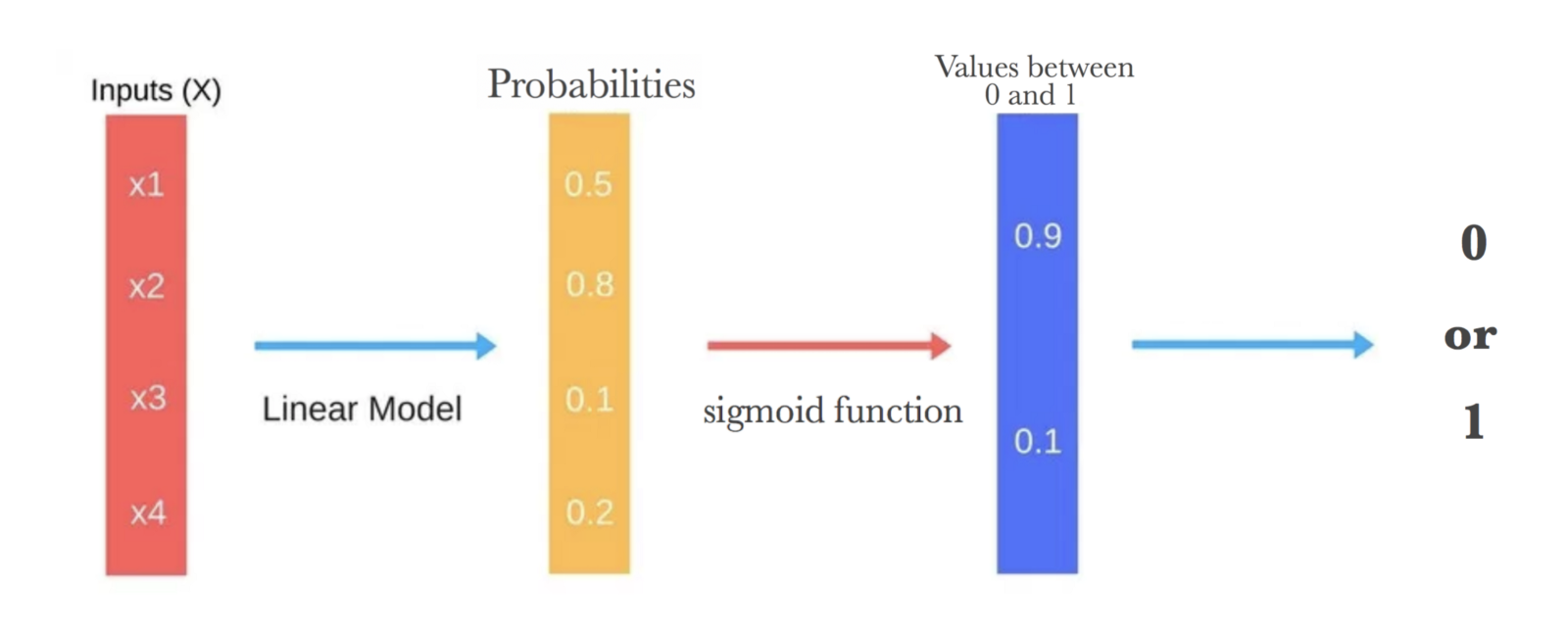
* What is Logistic Regression?

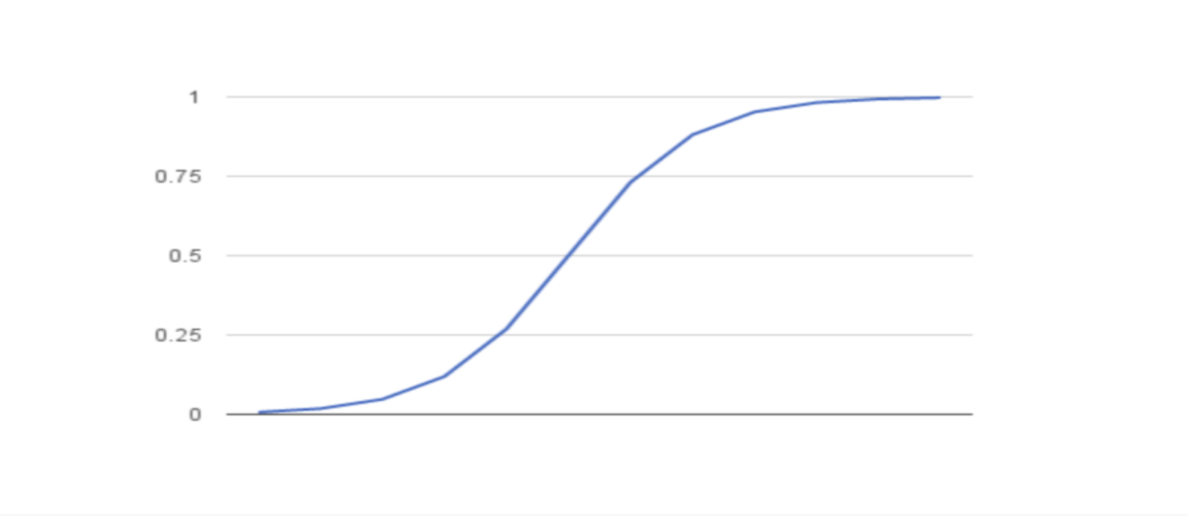
Like many other machine learning techniques, it is borrowed from the field of statistics and despite its name, it is not an algorithm for regression problems, where you want to predict a continuous outcome. Instead, Logistic Regression is the go-to method for binary classification. It gives you a discrete binary outcome between 0 and 1. To say it in simpler words, it’s outcome is either one thing or another. A simple example of a Logistic Regression problem would be an algorithm used for cancer detection that takes screening picture as an input and should tell if a patient has cancer (1) or not (0).

* How it works

Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it’s underlying logistic function.These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits. This values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.

The picture below illustrates the steps that logistic regression goes through to give you your desired output.



Below you can see how the logistic function (sigmoid function) looks like:

We want to maximize the likelihood that a random data point gets classified correctly, which is called Maximum Likelihood Estimation. Maximum Likelihood Estimation is a general approach to estimating parameters in statistical models. You can maximize the likelihood using different methods like an optimization algorithm. Newton’s Method is such an algorithm and can be used to find maximum (or minimum) of many different functions, including the likelihood function. Instead of Newton’s Method, you could also use Gradient Descent.

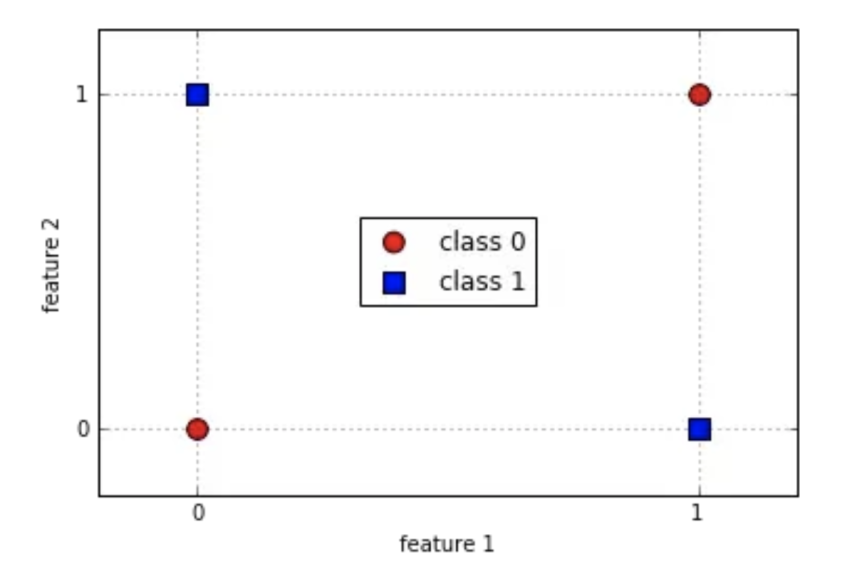
* Logistic VS. Linear Regression

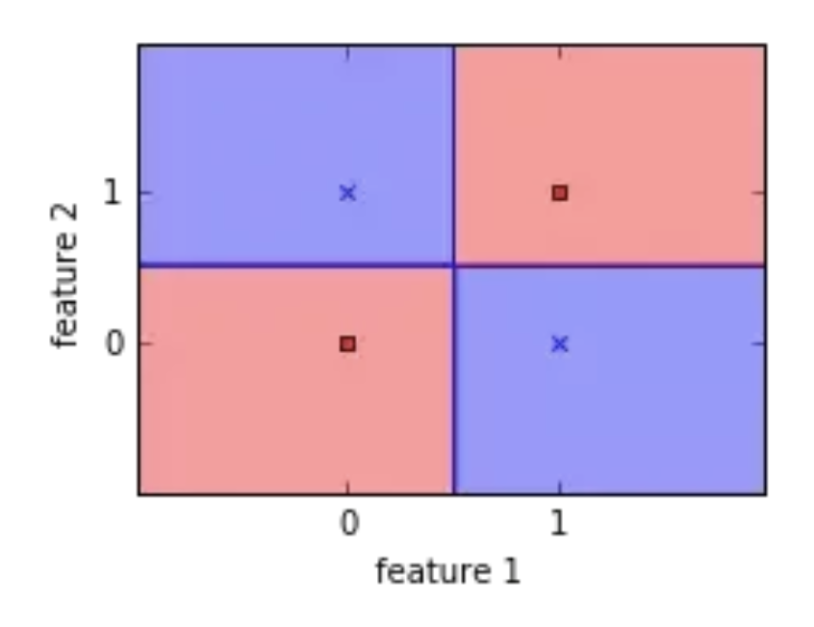
You may be asking yourself what the difference between logistic and linear regression is. Logistic regression gives you a discrete outcome but linear regression gives a continuous outcome. A good example of a continuous outcome would be a model that predicts the value of a house. That value will always be different based on parameters like it’s size or location. A discrete outcome will always be one thing (you have cancer) or another (you have no cancer).

### Advantages / Disadvantages

It is a widely used technique because it is very efficient, does not require too many computational resources, it’s highly interpretable, it doesn’t require input features to be scaled, it doesn’t require any tuning, it’s easy to regularize, and it outputs well-calibrated predicted probabilities.

Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic and also Linear Regression. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train. I typically start with a Logistic Regression model as a benchmark and try using more complex algorithms from there on.Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is also a good baseline that you can use to measure the performance of other more complex Algorithms. A disadvantage of it is that we can’t solve non-linear problems with logistic regression since it’s decision surface is linear. Just take a look at the example below that has 2 binary features from 2 examples.

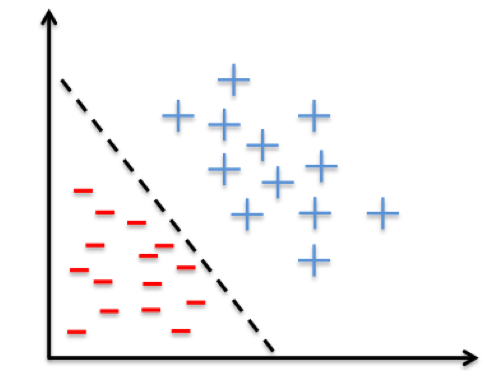


It is clearly visible that we can’t draw a line that separates these 2 classes without a huge error. To use a simple decision tree would be a much better choice.

Logistic Regression is also not one of the most powerful algorithms out there and can be easily outperformed by more complex ones. Another disadvantage is its high reliance on a proper presentation of your data. This means that logistic regression is not a useful tool unless you have already identified all the important independent variables. Since its outcome is discrete, Logistic Regression can only predict a categorical outcome. It is also an Algorithm that is known for its vulnerability to overfitting.

* When to use it

Like I already mentioned, Logistic Regression separates your input into two „regions” by a linear boundary, one for each class. Therefore it is required that your data is linearly separable, like the data points in the image below:



In other words: You should think about using logistic regression when your Y variable takes on only two values (e.g when you are facing a classification problem). Note that you could also use Logistic Regression for multiclass classification, which will be discussed in the next section.

* Summary

In this post, you have learned what Logistic Regression is and how it works. You now have a solid understanding of its advantages and disadvantages and know when you can use it. Also, you have discovered ways to use Logistic Regression to do multiclass classification with sklearn and why it is a good baseline to compare other Machine Learning algorithms with.

**Chapter – 4**

Libraries Used

4.1 Numpy

 is a library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices](https://en.wikipedia.org/wiki/Matrix_(math)), along with a large collection of [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [mathematical](https://en.wikipedia.org/wiki/Mathematics) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin)with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors.

4.2 Pandas

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), **pandas** is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license).[[2]](https://en.wikipedia.org/wiki/Pandas_(software)#cite_note-2) The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for data sets that include observations over multiple time periods for the same individuals.

## Library features

* DataFrame 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 subsetting 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[[3]](https://en.wikipedia.org/wiki/Pandas_(software)#cite_note-3) and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.

The library is highly optimized for performance, with critical code paths written in [Cython](https://en.wikipedia.org/wiki/Cython) or [C](https://en.wikipedia.org/wiki/C_(programming_language)).

4.3 MatPlotLib:

**Matplotlib** is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy). It provides an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [API](https://en.wikipedia.org/wiki/API) for embedding plots into applications using general-purpose [GUI toolkits](https://en.wikipedia.org/wiki/GUI_toolkit) like [Tkinter](https://en.wikipedia.org/wiki/Tkinter), [wxPython](https://en.wikipedia.org/wiki/WxPython), [Qt](https://en.wikipedia.org/wiki/Qt_(software)), or [GTK+](https://en.wikipedia.org/wiki/GTK%2B). There is also a [procedural](https://en.wikipedia.org/wiki/Procedural_programming) "pylab" interface based on a [state machine](https://en.wikipedia.org/wiki/State_machine) (like [OpenGL](https://en.wikipedia.org/wiki/OpenGL)), designed to closely resemble that of [MATLAB](https://en.wikipedia.org/wiki/MATLAB), though its use is discouraged.[[2]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-2) [SciPy](https://en.wikipedia.org/wiki/SciPy) makes use of matplotlib.

Matplotlib was originally written by [John D. Hunter](https://en.wikipedia.org/wiki/John_D._Hunter), as an active development community,[[3]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-3) and is distributed under a [BSD-style license](https://en.wikipedia.org/wiki/BSD_licenses). Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in 2012,[[4]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-4) and further joined by Thomas Caswell[[5]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-6)

As of 23 June 2017, matplotlib 2.0.x supports Python versions 2.7 through 3.6. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of matplotlib to support Python 2.6.[[7]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-7)

Matplotlib has pledge to not support Python 2 past 2020 by signing the [Python 3 Statement](https://python3statement.org/).

4.4 Seaborn:

Seaborn is a library for making attractive and informative statistical graphics in Python. It is built on top of [matplotlib](http://matplotlib.org/) and tightly integrated with the [PyData](http://pydata.org/) stack, including support for [numpy](http://www.numpy.org/) and [pandas](http://pandas.pydata.org/) data structures and statistical routines from [scipy](http://scipy.org/) and [statsmodels](http://statsmodels.sourceforge.net/).

Some of the features that seaborn offers are

* Several [built-in themes](https://seaborn.pydata.org/tutorial/aesthetics.html#aesthetics-tutorial) for styling matplotlib graphics
* Tools for choosing [color palettes](https://seaborn.pydata.org/tutorial/color_palettes.html#palette-tutorial) to make beautiful plots that reveal patterns in your data
* Functions for visualizing [univariate](https://seaborn.pydata.org/examples/distplot_options.html#distplot-options) and [bivariate](https://seaborn.pydata.org/examples/joint_kde.html#joint-kde) distributions or for [comparing](https://seaborn.pydata.org/examples/grouped_violinplots.html#grouped-violinplots) them between subsets of data
* Tools that fit and visualize [linear regression](https://seaborn.pydata.org/examples/anscombes_quartet.html#anscombes-quartet) models for different kinds of [independent](https://seaborn.pydata.org/examples/pointplot_anova.html#pointplot-anova) and [dependent](https://seaborn.pydata.org/examples/logistic_regression.html#logistic-regression) variables
* Functions that visualize [matrices of data](https://seaborn.pydata.org/examples/heatmap_annotation.html#heatmap-annotation) and use clustering algorithms to [discover structure](https://seaborn.pydata.org/examples/structured_heatmap.html#structured-heatmap) in those matrices
* A function to plot [statistical timeseries](https://seaborn.pydata.org/examples/timeseries_from_dataframe.html#timeseries-from-dataframe) data with flexible estimation and representation of uncertainty around the estimate
* High-level abstractions for structuring [grids of plots](https://seaborn.pydata.org/examples/faceted_histogram.html#faceted-histogram) that let you easily build [complex](https://seaborn.pydata.org/examples/many_facets.html#many-facets) visualizations

Seaborn aims to make visualization a central part of exploring and understanding data. The plotting functions operate on dataframes and arrays containing a whole dataset and internally perform the necessary aggregation and statistical model-fitting to produce informative plots. If matplotlib “tries to make easy things easy and hard things possible”, seaborn tries to make a well-defined set of hard things easy too.

The plotting functions try to do something useful when called with a minimal set of arguments, and they expose a number of customizable options through additional parameters. Some of the functions plot directly into a matplotlib axes object, while others operate on an entire figure and produce plots with several panels. In the latter case, the plot is drawn using a Grid object that links the structure of the figure to the structure of the dataset.

4.5 OS

The OS module in python provides functions for interacting with the operating system. OS, comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

**Chapter – 5**

Function Used

5.1 read\_csv( ):

this function helps us to create a dataframe by reading and storing the data from the given file that we make the system read.

5.2 .head( ):

This functions returns the first n number of rows of the dataframe.

5.3 .info( ):

This function gives us the entire information of the dataframe.

5.4 .describe( ):

This function generates various statistical values including NaN values.

5.5 .plot( )

This function is used to plot a continuous graph over the given variables and axises.

5.6 .subplot( ) :

This function is used to plot a discrete graph over the given variables and axises.

5.7 .figure( ):

This gives the user the ability to change the figure of the variables being plotted.

5.8 train\_test\_split:

This function allows us to convert the entire data stored into columns to be splitted into two categories i.e Train and Test, their size depending upon the percentage given.

5.9 .ravel( ):

This functions converts entire data into a single row.

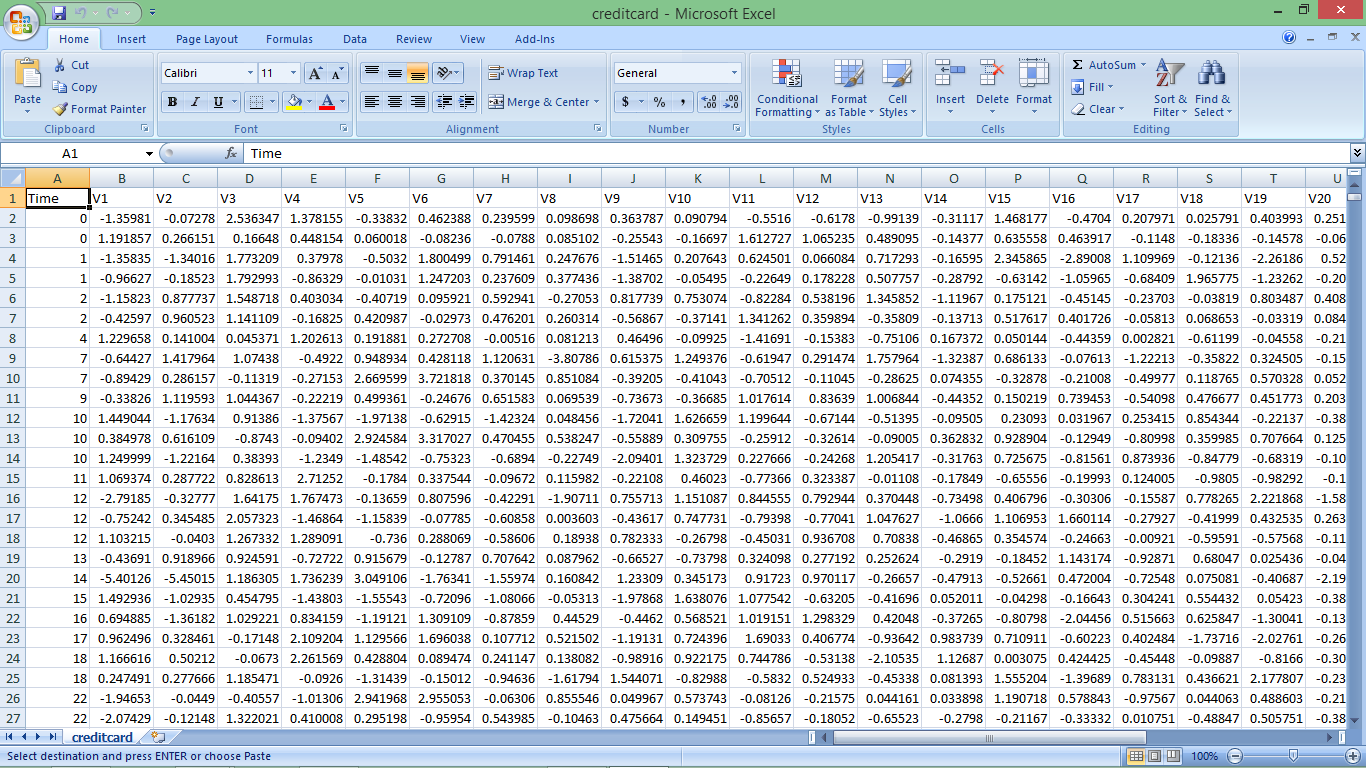
5.10 ..show( ):

This function is used to display the plotted contents.

**Chapter – 5**

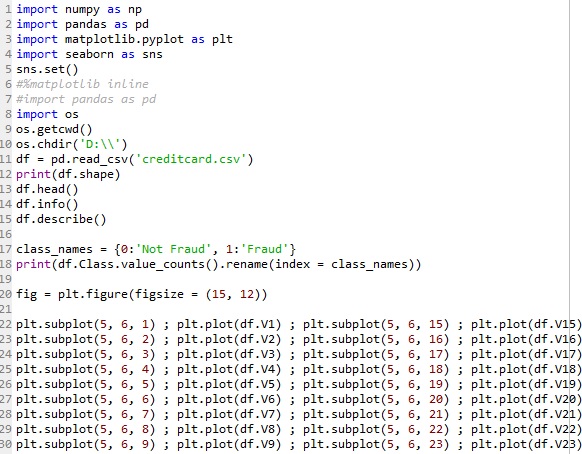
DataSet

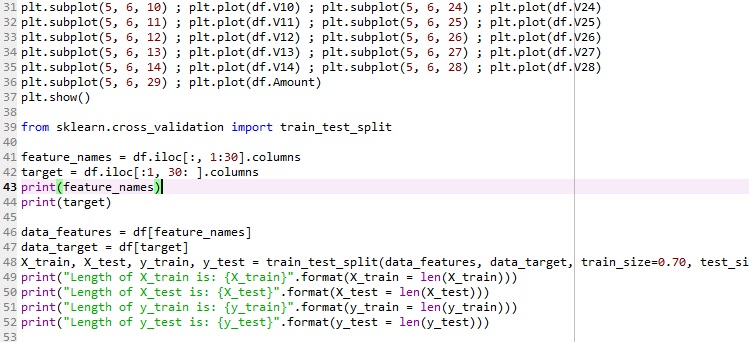
The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

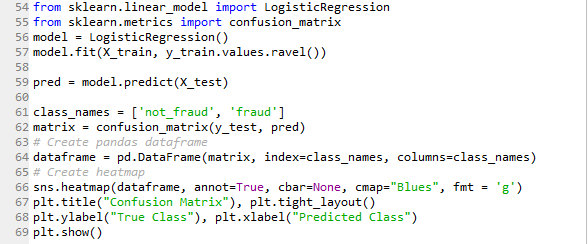


**Chapter – 6**

Code Window

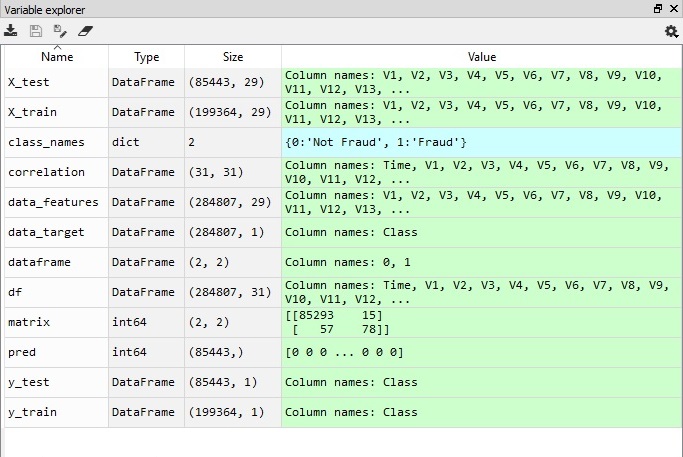






**Chapter – 6**

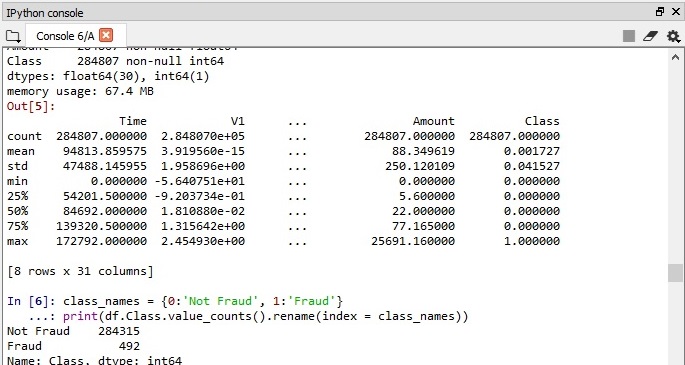
Variable Explorer



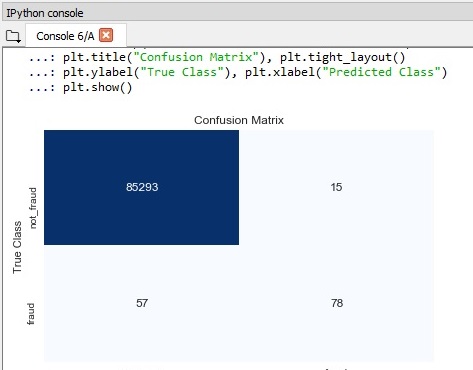
**Chapter – 7**

IPython Console

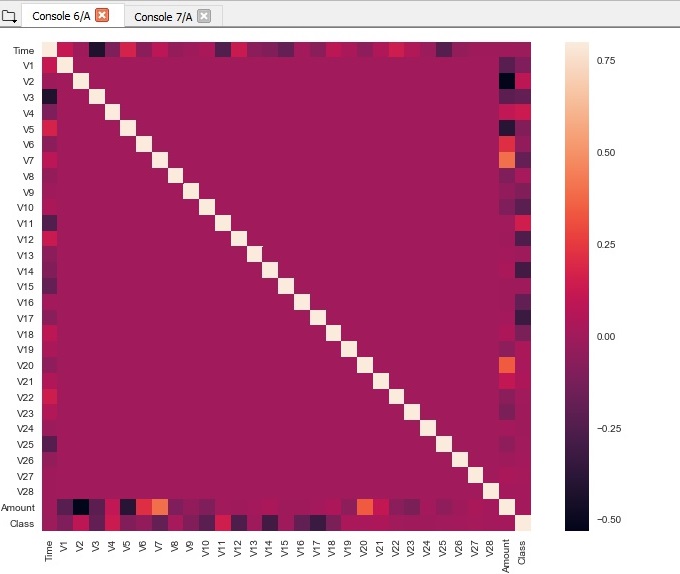
CLASS OUTPUT



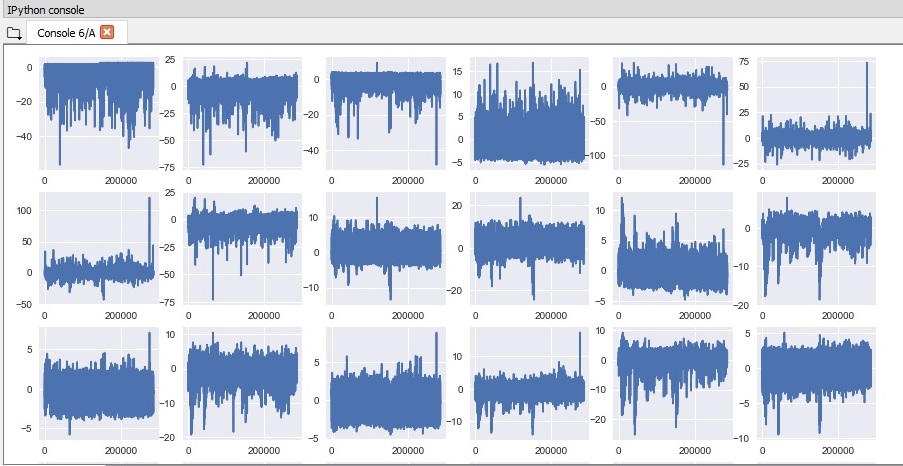
CONFUSION MATRIX



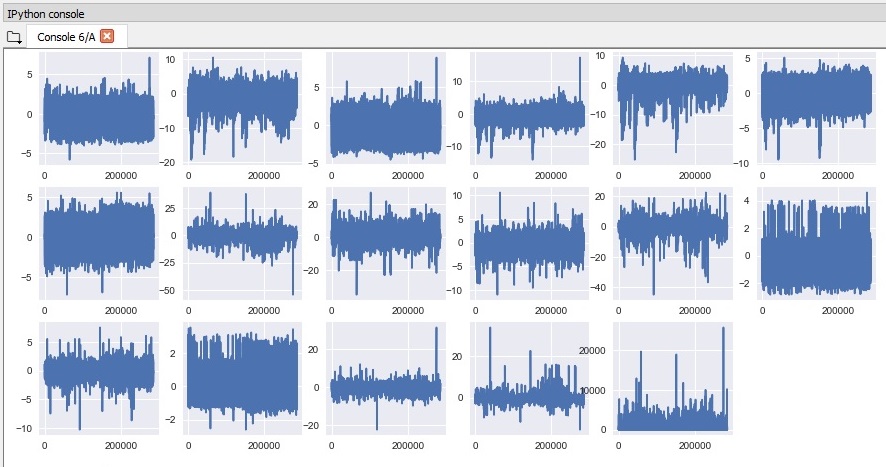
CORRELATION MATRIX



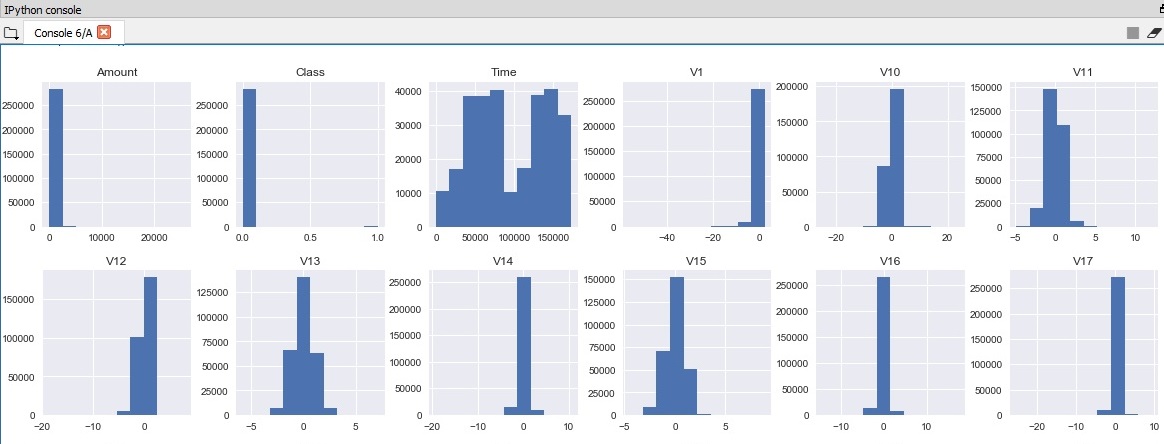
FEATURE PLOT 1



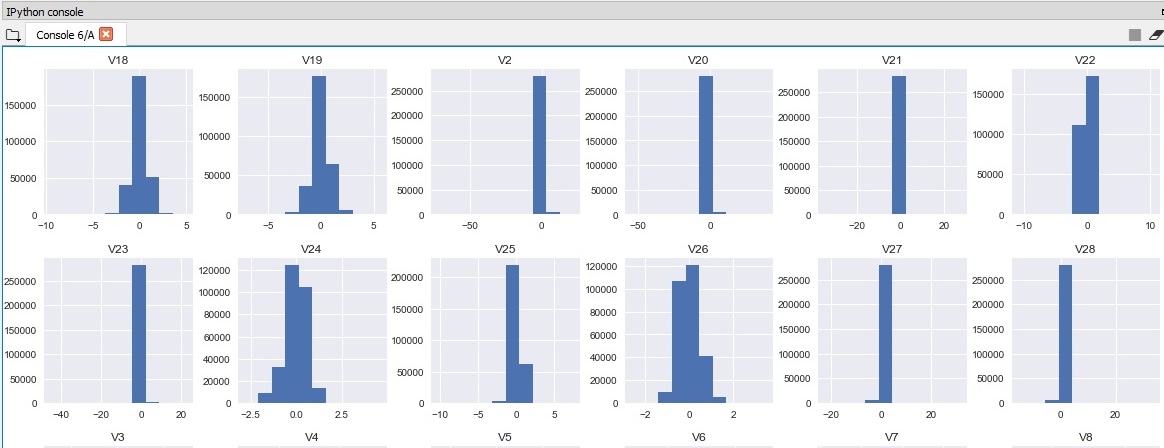
FEATURE PLOT 2



HISTOGRAM 1



HISTOGRAM 2



HISTOGRAM 3



**Chapter – 8**

Code

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

#%matplotlib inline

#import pandas as pd

import os

os.getcwd()

os.chdir('D:\\')

df = pd.read\_csv('creditcard.csv')

print(df.shape)

df.head()

df.info()

df.describe()

class\_names = {0:'Not Fraud', 1:'Fraud'}

print(df.Class.value\_counts().rename(index = class\_names))

fig = plt.figure(figsize = (15, 12))

plt.subplot(5, 6, 1) ; plt.plot(df.V1) ; plt.subplot(5, 6, 15) ; plt.plot(df.V15)

plt.subplot(5, 6, 2) ; plt.plot(df.V2) ; plt.subplot(5, 6, 16) ; plt.plot(df.V16)

plt.subplot(5, 6, 3) ; plt.plot(df.V3) ; plt.subplot(5, 6, 17) ; plt.plot(df.V17)

plt.subplot(5, 6, 4) ; plt.plot(df.V4) ; plt.subplot(5, 6, 18) ; plt.plot(df.V18)

plt.subplot(5, 6, 5) ; plt.plot(df.V5) ; plt.subplot(5, 6, 19) ; plt.plot(df.V19)

plt.subplot(5, 6, 6) ; plt.plot(df.V6) ; plt.subplot(5, 6, 20) ; plt.plot(df.V20)

plt.subplot(5, 6, 7) ; plt.plot(df.V7) ; plt.subplot(5, 6, 21) ; plt.plot(df.V21)

plt.subplot(5, 6, 8) ; plt.plot(df.V8) ; plt.subplot(5, 6, 22) ; plt.plot(df.V22)

plt.subplot(5, 6, 9) ; plt.plot(df.V9) ; plt.subplot(5, 6, 23) ; plt.plot(df.V23)

plt.subplot(5, 6, 10) ; plt.plot(df.V10) ; plt.subplot(5, 6, 24) ; plt.plot(df.V24)

plt.subplot(5, 6, 11) ; plt.plot(df.V11) ; plt.subplot(5, 6, 25) ; plt.plot(df.V25)

plt.subplot(5, 6, 12) ; plt.plot(df.V12) ; plt.subplot(5, 6, 26) ; plt.plot(df.V26)

plt.subplot(5, 6, 13) ; plt.plot(df.V13) ; plt.subplot(5, 6, 27) ; plt.plot(df.V27)

plt.subplot(5, 6, 14) ; plt.plot(df.V14) ; plt.subplot(5, 6, 28) ; plt.plot(df.V28)

plt.subplot(5, 6, 29) ; plt.plot(df.Amount)

plt.show()

from sklearn.cross\_validation import train\_test\_split

feature\_names = df.iloc[:, 1:30].columns

target = df.iloc[:1, 30: ].columns

print(feature\_names)

print(target)

data\_features = df[feature\_names]

data\_target = df[target]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data\_features, data\_target, train\_size=0.70, test\_size=0.30, random\_state=1)

print("Length of X\_train is: {X\_train}".format(X\_train = len(X\_train)))

print("Length of X\_test is: {X\_test}".format(X\_test = len(X\_test)))

print("Length of y\_train is: {y\_train}".format(y\_train = len(y\_train)))

print("Length of y\_test is: {y\_test}".format(y\_test = len(y\_test)))

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

model = LogisticRegression()

model.fit(X\_train, y\_train.values.ravel())

pred = model.predict(X\_test)

class\_names = ['not\_fraud', 'fraud']

matrix = confusion\_matrix(y\_test, pred)

# Create pandas dataframe

dataframe = pd.DataFrame(matrix, index=class\_names, columns=class\_names)

# Create heatmap

sns.heatmap(dataframe, annot=True, cbar=None, cmap="Blues", fmt = 'g')

plt.title("Confusion Matrix"), plt.tight\_layout()

plt.ylabel("True Class"), plt.xlabel("Predicted Class")

plt.show()

**Chapter – 9**

User Interface

Code :

Block I

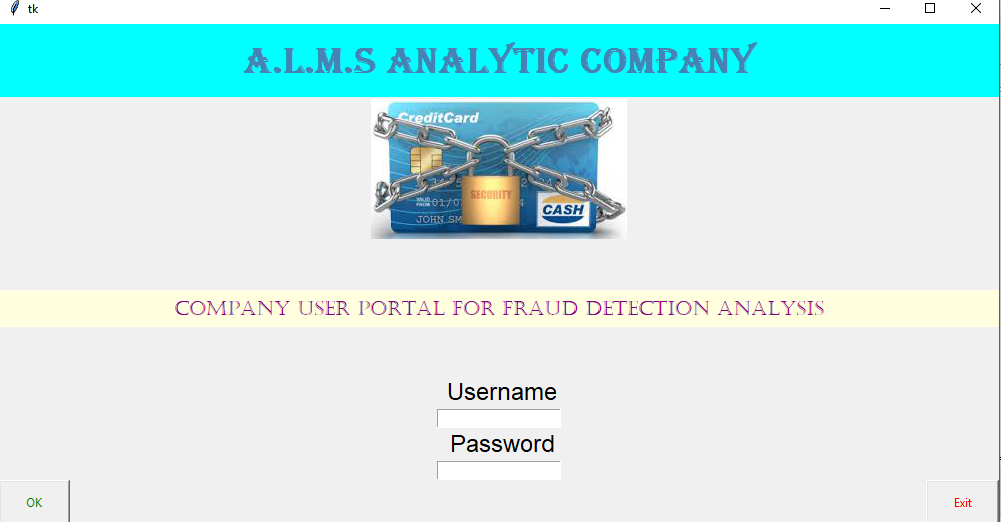
**import** tkinter  
**import** user\_test\_2  
**import** tkinter.messagebox  
**from** tkinter **import** \*  
  
**def** trial():  
 user\_test\_2.mainFrame()  
 **return**root = tkinter.Tk()  
  
geomet = root.geometry(**'1000x500+0+0'**)  
  
head\_title = Label(root, font=(**'algerian'**, 28, **'bold'**), text=**" A.L.M.S Analytic Company "**,bg = **'cyan'**, fg=**"steel blue"**, bd=15,  
 anchor=**'n'**)  
head\_title.pack(fill = X ,side = **'top'**)  
  
photo = PhotoImage(file=**"credit\_card.png"**)  
label = Label(root, image=photo, anchor = **'n'**)  
label.pack(side = **'top'**)  
  
label\_enter = Label(root, font = (**'castellar'**,15), text = **" Company user portal for fraud detection analysis "**, bg = **'lightyellow'**, fg = **'purple'**,bd = 5, anchor = **'n'**)  
label\_enter.pack(expand = **True**, fill = X , side = **'top'** )  
  
label\_user = Label(root , font = (**'ariel'**,18), text = **" Username"**,fg = **'black'**,anchor = **'nw'**)  
entry\_user = Entry(root)  
label\_user.pack()  
entry\_user.pack()  
  
label\_pass = Label(root , font = (**'ariel'**,18), text = **" Password"**,fg = **'black'**,anchor = **'ne'**)  
entry\_pass = Entry(root)  
label\_pass.pack()  
entry\_pass.pack()  
  
but\_1 = Button(root, text = **" OK "**, padx=20 , pady = 10, fg = **'green'**, anchor = **'sw'**,command = trial )  
but\_1.pack(side = **'left'**)  
  
but\_2 = Button(root, text=**" Exit "**, fg=**'red'**, padx=20, pady = 10, command= root.quit, anchor=**'se'**)  
but\_2.pack(side = **'right'**)  
  
  
root.mainloop()

Block II

**import** hist\_credit  
**import** cormat\_credit  
**import** plot\_features  
**import** tkinter  
**from** tkinter **import** \*  
  
**def** trial\_1():  
 hist\_credit.hist\_disp()  
 **return  
def** trial\_2():  
 cormat\_credit.cormat\_disp()  
 **return  
def** trial\_3():  
 plot\_features.plot\_disp()  
 **return  
  
  
def** mainFrame():  
 root = tkinter.Tk()  
 root.wm\_title(**"Access Allowed"**)  
  
 but\_hist = Button(root, font = (**'arieal'**,18),text= **" Histrogram ( features vs Class) "**,bg = **'white'** , fg = **'lightcoral'**, anchor = **'n'**, command = trial\_1)  
 but\_hist.pack(side=**"top"**, fill= X, padx=10, pady=10)  
  
 but\_cormat = Button(root, font=(**'arieal'**, 18), text=**" Co-Relation ( features) "**, bg = **'white'** ,fg=**'blue'**, anchor=**'n'**,command = trial\_2)  
 but\_cormat.pack(side=**"top"**, fill= X, padx=10, pady=10)  
  
 but\_feature = Button(root, font=(**'arieal'**, 18), text=**" Features "**, bg = **'white'** ,fg=**'yellow'**, anchor=**'n'**,command = trial\_3)  
 but\_feature.pack(side=**"top"**, fill= X, padx=10, pady=10)  
  
 root.mainloop()

**OUTPUT**:

Main Frame Of User Panel



Second Frame

