

# **1. INTRODUCTION**

The increasing amount of online trading is attracting criminal activities. In online trading, the credit cards are usually used as virtual card. These online transactions are deceived by identification of the card details (e.g., card ID, secure code), generally the types of deception are classified into:

- i) Synthetic identity fraud is the use of probable but fictitious identities. These are trouble free to generate but harder to apply successfully.
- ii) Real identity fraud is to illicit use of innocent people's complete identity details. These can be harder to obtain, but easier to successfully apply.

In reality, identity crime can be devoted to a mix of both synthetic identity fraud and real identity fraud details. Identity crime has become famous because there is so much real identity data available such as Web and unsecured mailboxes. It has also become easy for hackers to bury their accurate identities which are enabling them to make more frauds. So a Credit Card Fraud Detection model is proposed to overcome this problem by fusing multiple algorithms and using Big Data technologies to the existing system. The developed system will be very useful for the customers as well as the merchants who are highly dependent on the Online Banking for their businesses.

## **EXISTING SYSTEM**

The computational capacity of the existing system is not sufficient due to the increasing amount of online transactions. The size of historical transactions can reach PBs and even EBs as number of transactions coming into system reach million per second. Scalability of the system is also at stake. The Visa Company could only analyse 2% of its historical transactions, and make one update of its detection model every 2 or 3 days.

## **PROPOSED SYSTEM**

The main objective of this system is to achieve the challenges of the drawbacks of existing system like scalability, high response time, efficiency, imbalance of data, false predictions etc. The proposed system uses big data technology to achieve scalability and also to handle large amounts of data effectively. Also the accuracy is being improved by fusing multiple detection models.

## **2. SYSTEM REQUIREMENT SPECIFICATION**

System analysis is the first technical step in software engineering process. It is at this point that a general statement of software scope is refined into concrete specification that becomes the foundation for all software engineering activities that follow. Analysis must focus on information, functional and behavioural domain of the problem. To better understand what is required, models are created and the problem is partitioned. In many cases it is not possible to completely specify a problem at an early stage.

### **2.1 System Requirement Specification (SRS)**

A software requirement specification is developed as a consequence of analysis. Review is essential to ensure that the developer and customers have the same perception.

Software Requirement Specification is the starting point of the software development activity. The Software Requirement Specification is produced at the culmination of the analysis task. The introduction of the software requirement specification states the goals and objectives of the software, describing it in the context of the computer-based system. The SRS includes an information description, functional description, behavioural description, validation criteria.

The purpose of this document is to present the software requirements in a precise and easily understood manner. This document provides the functional, performance, design and verification requirements of the software to be developed.

This is the only document that describes the requirements of the system. This is meant for use by the developers and will also be the basis for validating the final delivered system.

A requirement is a statement about what the proposed system will do that all stakeholders agree must be made true in order for the customer's problem to be adequately solved.

Requirements can be divided in two major types, functional and non-functional.

Domain analysis is the process by which a software engineer learns background information. He or she has to learn sufficient information so as to be able to understand the problem and make good decisions during requirement analysis and other stages of software engineering process.

To perform domain analysis, you must gather information from whatever sources of information are available: these include the domain experts; and any books about the domain, any existing software and its documentation, and any other documents you can find.

## **2.2. Functional Requirements**

The functional requirements of the system defines a function of software system or its components. A function is described a set of inputs, behaviour of a system and output.

The following are the functional requirements of proposed system

### **INPUT:**

Credit card Transaction data.

### **PROCESS:**

Transactions are passed to the proposed model which uses Logistic Regression, SVC and Naïve Bayesian Classifier. The fraud scores obtained by each model is merged into single score  $f(t)$  and checked with the threshold values ( $\theta_U$ -Lower bound of genuine transactions,  $\theta_L$ -Upper bound of fraud transactions). If  $f(t) > \theta_U$  then the transaction is genuine and if  $f(t) < \theta_L$  then the transaction is fraud. If  $\theta_L < f(t) < \theta_U$  then the transaction is passed to decision tree classifier for further classification.

### **OUTPUT:**

The prediction whether the transaction is genuine or fraud.

## **2.3. Non-Functional Requirements**

Non-functional requirements are constraints that must be adhered to during development. They limit what resources can be used and set bounds on aspects of the software's quality.

One of the most important things about non-functional requirements is to make them verifiable. The verification is normally done by measuring various aspects of the system and seeing if the measurements confirms to the requirements. Non-Functional Requirements are divided into several groups. The first group of categories reflects the five qualities attributes

- Usability
- Efficiency

- Reliability
- Maintainability
- Reusability

The second group of non-functional requirements categories constraints the environment and technology of the system

**1. Platform:** It is quite important to make it clear on what hardware and operating system of the software must work on. Normally such requirements specify the least powerful platforms and declare that it must work on anything more recent or more powerful.

**Hardware Requirements:**

- 4GB RAM
- 1.7-2.4 GHz processor speed
- 500 GB HDD

**2. Technology to be used:** While it is wise to give the designers as much flexibility as possible to choose how to implement the system, sometimes constraints must be imposed. Requirements are normally stated to ensure that all systems in an organization use the same technology – this reduces the need to train people in different technologies.

**Software Requirements:**

- Ubuntu 16.04
- Python 2.7
- Anaconda 5.0.1
- Java 1.8
- Hadoop 2.5.2

The third group of non-functional requirements categories constraint the project plan and development methods

- **Development process (methodology) to be used:** In order to ensure quality, some requirements documents specify that certain processes be followed; for example, particular approaches to testing. The details of the process should not be included in the requirements; instead a reference should be made to other documents that describe the process.

- **Cost and delivery date:** One of the biggest challenges in software engineering is accurately forecasting how much time and effort it will take either to develop a system or to make a specific set of changes. All software developers have to participate in cost estimation.

## **2.4 Feasibility Study**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time.

There are aspects in the feasibility study portion of the preliminary investigation.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

### **2.4.1 Technical Feasibility**

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

1. Analyzes the technical skills and capabilities of the software development team members
2. Determines whether the relevant technology is stable and established
3. Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

### **2.4.2 Operational Feasibility**

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority
- Determines whether the solution suggested by the software development team is acceptable
- Analyzes whether users will adapt to a new software
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

### **2.4.3 Economic Feasibility**

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization
- Cost required conducting full software investigation (such as requirements elicitation and requirements analysis)
- Cost of hardware, software, development team, and training.

## 3. DESIGN

### 3.1. UML DIAGRAMS

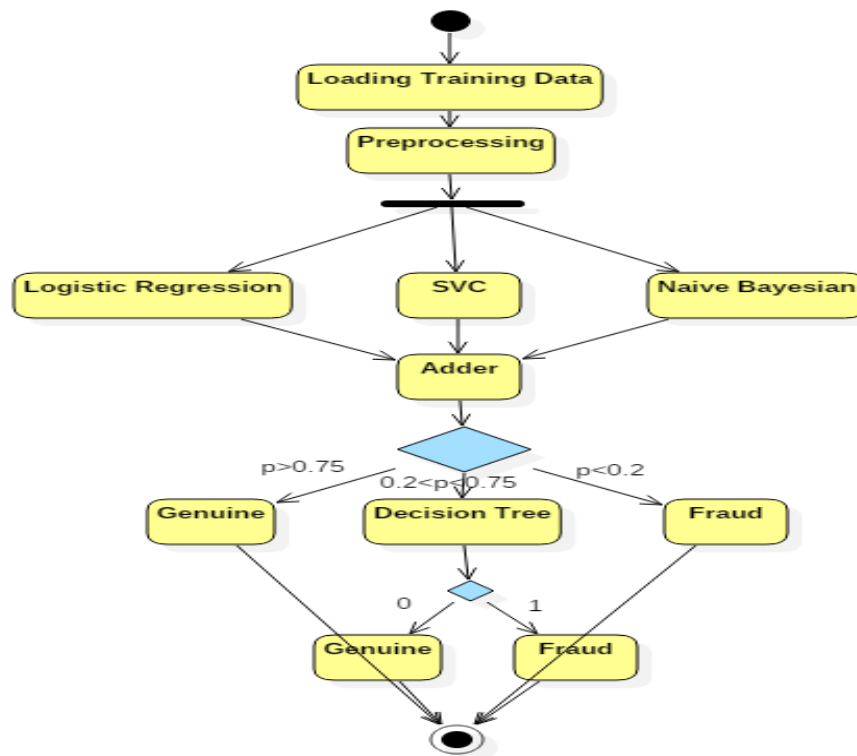
The Unified Modelling Language (UML) is a general-purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. It allows software to be visualised in multiple dimension, so that a computer system can be completely understood before construction begins.

Conceptual model of UML can be mastered by learning the following three major elements:

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

#### 3.1.1 ACTIVITY DIAGRAM

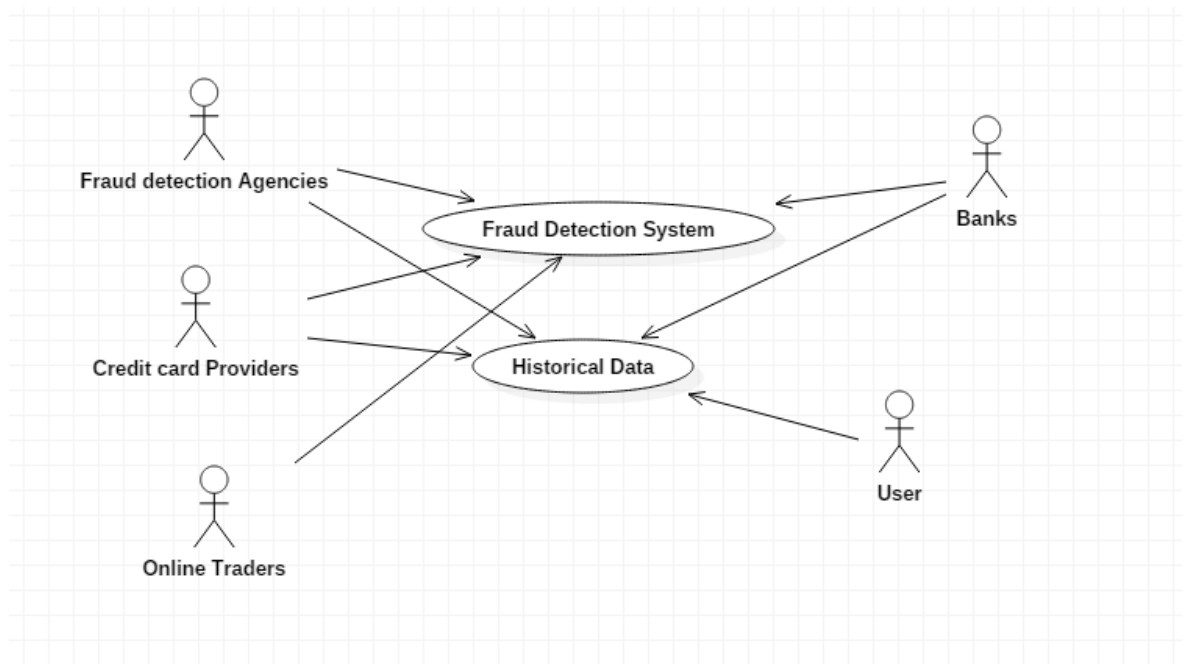
Activity diagrams illustrate the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system.



### 3.1.2 USECASE DIAGRAM

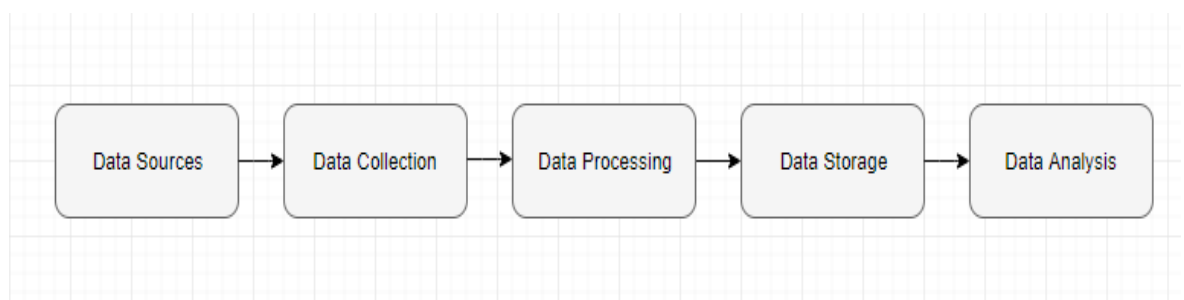
To model a system the most important aspect is to capture the dynamic behavior.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.



### 3.1.3 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system, modelling its process aspects. It is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. It can also be used for the visualization of data processing. A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored.





## 3.2 ALGORITHMS AND METHODOLOGY

### 3.2.1 LOGISTIC REGRESSION

It is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable. The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a *logit transformation* of the probability of presence of the characteristic of interest:

$$\text{logit}(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_kX_k$$

where  $p$  is the probability of presence of the characteristic of interest,  $X_i$  are attributes and  $b_i$  are values of coefficient of constants. The logit transformation is defined as the logged odds:

$$\text{odds} = \frac{p}{1-p} = \frac{\text{probability of presence of characteristic}}{\text{probability of absence of characteristic}}$$

and

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

### 3.2.2 BAYESIAN CLASSIFIER

It is a classification technique based on Bayes Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'. A Gaussian Naive Bayes algorithm is a special type of NB algorithm. It's specifically used when the features have continuous

values. It's also assumed that all the features are following a gaussian distribution i.e., normal distribution.

Bayesian Equation:

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

Likelihood
Class Prior Probability

Posterior Probability
Predictor Prior Probability

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Gaussian Bayesian Equation:

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Process:

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
All	5	9
	=5/14	=9/14
	0.36	0.64

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

**Problem:** Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

$$P(\text{Yes} | \text{Sunny}) = P(\text{Sunny} | \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

Here we have  $P(\text{Sunny} | \text{Yes}) = 3/9 = 0.33$ ,  $P(\text{Sunny}) = 5/14 = 0.36$ ,  $P(\text{Yes}) = 9/14 = 0.64$

Now,  $P(\text{Yes} | \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$ , which has higher probability.

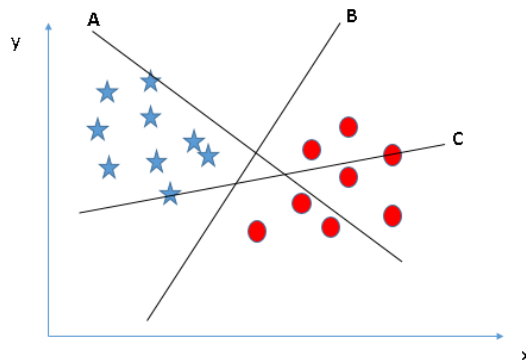
Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

### 3.2.3 SUPPORT VECTOR MACHINES (SVM)

It is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. In other words, given labelled training data (*supervised learning*), the algorithm outputs an optimal hyper plane which categorizes new examples. In two-dimensional space this hyper plane is a line dividing a plane in two parts where in each class lay in either side. The distance between the hyper plane and the nearest data point from either set is known as the margin. The goal is to choose a hyper plane with the greatest possible margin between the hyper plane and any point within the training set, giving a greater chance of new data being classified correctly. It works well on smaller cleaner datasets.

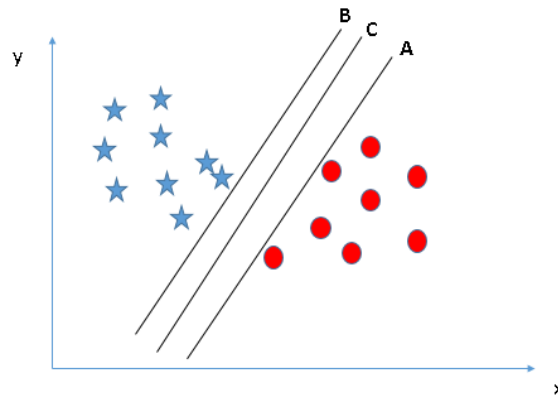
Process:

- Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



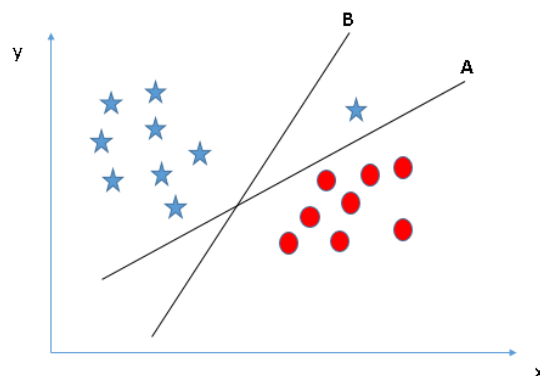
Thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

- Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

- Identify the right hyper-plane (Scenario-3)



Here SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A**.

### 3.2.4 DECISION TREES

It is a type of supervised learning algorithm that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets based on most significant splitter or differentiator in input variables. A tree can be learned by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. The decision of making strategic splits heavily affects a tree's accuracy. Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

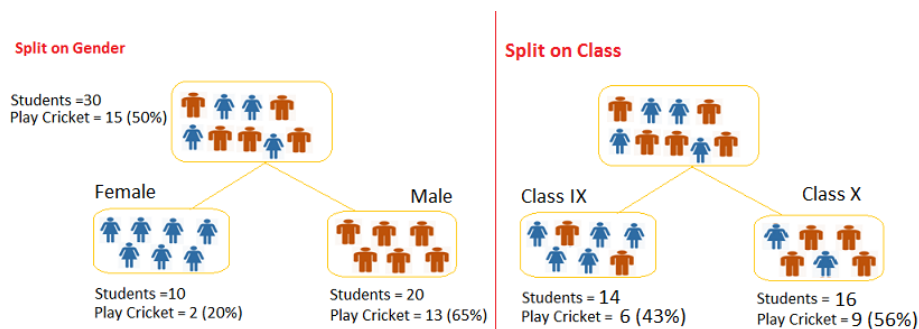
Steps to Calculate Gini for a split

1. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure ( $p^2 + q^2$ ).
2. Calculate Gini for split using weighted Gini score of each node of that split.

The attribute that has higher Gini index is splinted first.

Process:

Referring to example used above, where we want to segregate the students based on target variable ( playing cricket or not ). In the snapshot below, we split the population using two input variables Gender and Class. Now, I want to identify which split is producing more homogeneous sub-nodes using Gini index.



### Split on Gender:

1. Calculate, Gini for sub-node Female =  $(0.2)*(0.2)+(0.8)*(0.8)=0.68$
2. Gini for sub-node Male =  $(0.65)*(0.65)+(0.35)*(0.35)=0.55$
3. Calculate weighted Gini for Split Gender =  $(10/30)*0.68+(20/30)*0.55 = \mathbf{0.59}$

### Similar for Split on Class:

1. Gini for sub-node Class IX =  $(0.43)*(0.43)+(0.57)*(0.57)=0.51$
2. Gini for sub-node Class X =  $(0.56)*(0.56)+(0.44)*(0.44)=0.51$
3. Calculate weighted Gini for Split Class =  $(14/30)*0.51+(16/30)*0.51 = \mathbf{0.51}$

Here Gini score for Split on Gender is higher than Split on Class, hence, the node split will take place on Gender.

### 3.2.5 DATASET

The datasets contain transactions made by credit cards in September 2013 by European cardholders which was taken from KAGGLE. 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. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of distinct principal components is equal to the smaller of the number of original variables or the number of observations minus one. 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-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

### 3.2.6 CONFUSION MATRIX

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

True Positive (TP) : Observation is positive, and is predicted to be positive.

False Negative (FN) : Observation is positive, but is predicted negative.

True Negative (TN) : Observation is negative, and is predicted to be negative.

False Positive (FP) : Observation is negative, but is predicted positive.

Measure	Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Misclassification rate (1 – Accuracy)	$\frac{FP + FN}{TP + TN + FP + FN}$
Sensitivity (or Recall)	$\frac{TP}{TP + FN}$
Specificity	$\frac{TN}{TN + FP}$
Precision (or Positive Predictive Value)	$\frac{TP}{TP + FP}$

## 4. DEVELOPMENT

### 4.1 SAMPLE CODE

#### Pre-processing

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.gridspec as gridspec

df=pd.read_csv("creditcard.csv")
attr_list=['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
          'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
          'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount']
for v in attr_list:
    df[v].fillna(df[v].mean(), inplace=True)
x = df[['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
        'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
        'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount']]
y = df['Class']
fig, ax = plt.subplots(1, 1)
ax.pie(df.Class.value_counts(),autopct='%1.1f%%', labels=['Genuine','Fraud'],
      colors=['y','r'])
plt.axis('equal')
plt.ylabel("")
plt.show()
plt.figure(figsize=(6,28*4))
for i, col in enumerate(df[df.iloc[:,0:29].columns]):
    sns.distplot(df[col][df.Class == 1], bins=50, color='r')
    sns.distplot(df[col][df.Class == 0], bins=50, color='g')
    plt.xlabel("")
    plt.title('feature: ' + str(col))
    plt.show()
```



### **Getting Data from HDFS**

```
# import the python subprocess module
import subprocess

def run_cmd(args_list):
    print('Running system command: {0}'.format(' '.join(args_list)))
    proc = subprocess.Popen(args_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    s_output, s_err = proc.communicate()
    s_return = proc.returncode
    return s_return, s_output, s_err

(ret, out, err)= run_cmd(['hdfs', 'dfs', '-get', '/user', 'home/sampath/Desktop/project'])

hdfs_file_path = '/user/samput'
cmd = ['hdfs', 'dfs', '-test', '-d', hdfs_file_path]
ret, out, err = run_cmd(cmd)
```

### **Training and Prediction:**

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import itertools

#reading data
data= pd.read_csv("creditcard.csv")
```

```

sc= StandardScaler()
data["scaled_Amount"]=sc.fit_transform(data["Amount"].values.reshape(-1,1))
#dropping time and old amount column
data=
data.drop(["Time","Amount","V8","V13","V15","V20","V22","V23","V24","V25","V26",
"V27","V28"], axis= 1)
f_cnt= len(data[data["Class"]==1])
n_cnt = len(data[data["Class"]==0])
#getting fraud and normal transaction indexes
fraud_index= np.array(data[data["Class"]==1].index)
normal_index= data[data["Class"]==0].index
#choosing random normal indices equal to the number of fraudulent transactions
ind= np.random.choice(normal_index, f_cnt, replace= False)
ind= np.array(ind)
# concatenate fraud index and normal index to create a list of indices
ui= np.concatenate([fraud_index, ind])
#use the undersampled indices to build the ud dataframe
ud= data.iloc[ui, :]
X= ud.iloc[:, ud.columns != "Class"].values
y= ud.iloc[:, ud.columns == "Class"].values
X_train_u, X_test_u, y_train_u, y_test_u = train_test_split(X, y, test_size= 0.25,
random_state= 0)
ud["scaled_Amount"]= sc.fit_transform(ud.iloc[:,18].values.reshape(-1,1))
#separating the x and y variables to fit our model
X_full= data.iloc[:, ud.columns != "Class"].values
y_full= data.iloc[:, ud.columns == "Class"].values
X_train, X_test, y_train, y_test = train_test_split(X_full, y_full, test_size= 0.25,
random_state= 0)
lr = LogisticRegression(class_weight="balanced")
lr.fit(X_train, y_train.ravel())
y_pred_lr=lr.predict(X_test)
y_pred_prob_lr = lr.predict_proba(X_test)

```

```

dt = tree.DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt=lr.predict(X_test)
y_pred_prob_dt=dt.predict_proba(X_test)
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_gnb=gnb.predict(X_test)
y_pred_prob_gnb =gnb.predict_proba(X_test)
svc = SVC(C= 1, kernel="linear", random_state= 0,probability=True)
svc.fit(X_train_u, y_train_u.ravel())
y_pred_svc= svc.predict(X_test)
y_pred_prob_svc=svc.predict_proba(X_test)
y_pred=[]
l=len(y_pred_prob_lr)
for i in range(l):
    m=(3*y_pred_prob_lr[i][0]+2*y_pred_prob_gnb[i][0]+3*y_pred_prob_svc[i][0])/8
    if m>0.75:
        y_pred.append(0)
        f1.write(', '.join(map(str,X_test[i]))+",0" + "\n")
    elif m<0.2:
        y_pred.append(1)
        f0.write(', '.join(map(str,X_test[i])) + ",1" + "\n")
    else:
        r = X_test[i]
        v=dt.predict(r.reshape(1,-1))
        y_pred.append(v)
        if v==0:
            f1.write(', '.join(map(str,X_test[i]))+",0" + "\n")
        else:
            f0.write(', '.join(map(str,X_test[i])) + ",1" + "\n")

```

**Post-processing:**

```
CM = metrics.confusion_matrix(y_test, y_pred)
print "Confusion Martix\n",CM
acc=float((CM[1][1]+CM[0][0])/(CM[0][0] + CM[0][1]+CM[1][0] + CM[1][1])*100
prec=float(CM[0][0])/(CM[0][0]+CM[1][0])*100
rec=float(CM[0][0])/(CM[0][0]+CM[0][1])*100
spec=float(CM[1][1])/(CM[1][0]+CM[1][1])*100
print "Accuracy =",acc
print "Precision =",prec
print "Recall =",rec
print "Specificity =",spec
def plot_confusion_matrix(cm, classes):
    plt.imshow(cm, cmap=plt.cm.Blues)
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],horizontalalignment="center",color="black")
    plt.show()
plt.figure()
plot_confusion_matrix(CM, classes="0,1")
```

**Writing Fraud Data to HDFS:**

```
import subprocess
def run_cmd(args_list):
    print('Running system command: {0}'.format(' '.join(args_list)))
    proc = subprocess.Popen(args_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    s_output, s_err = proc.communicate()
    s_return = proc.returncode
    return s_return, s_output, s_err

(ret, out, err)= run_cmd(['hdfs', 'dfs', '-put', 'fraud', '/user'])
```

```
hdfs_file_path = '/user/sampu'
cmd = ['hdfs', 'dfs', '-test', '-d', hdfs_file_path]
ret, out, err = run_cmd(cmd)
```

### Writing Genuine Data to HDFS:

```
import subprocess
def run_cmd(args_list):
    print('Running system command: {0}'.format(' '.join(args_list)))
    proc = subprocess.Popen(args_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    s_output, s_err = proc.communicate()
    s_return = proc.returncode
    return s_return, s_output, s_err
```

```
(ret, out, err)= run_cmd(['hdfs', 'dfs', '-put', 'genuine', '/user'])
```

```
hdfs_file_path = '/user/sampu'
cmd = ['hdfs', 'dfs', '-test', '-d', hdfs_file_path]
ret, out, err = run_cmd(cmd)
```

### 4.2 Screenshots:

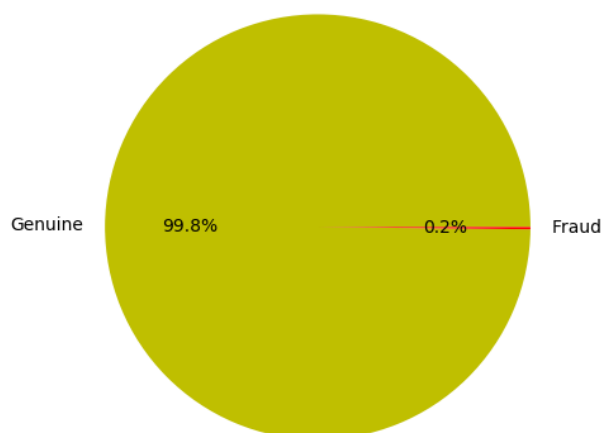


Fig 4.3.1 Class Distribution

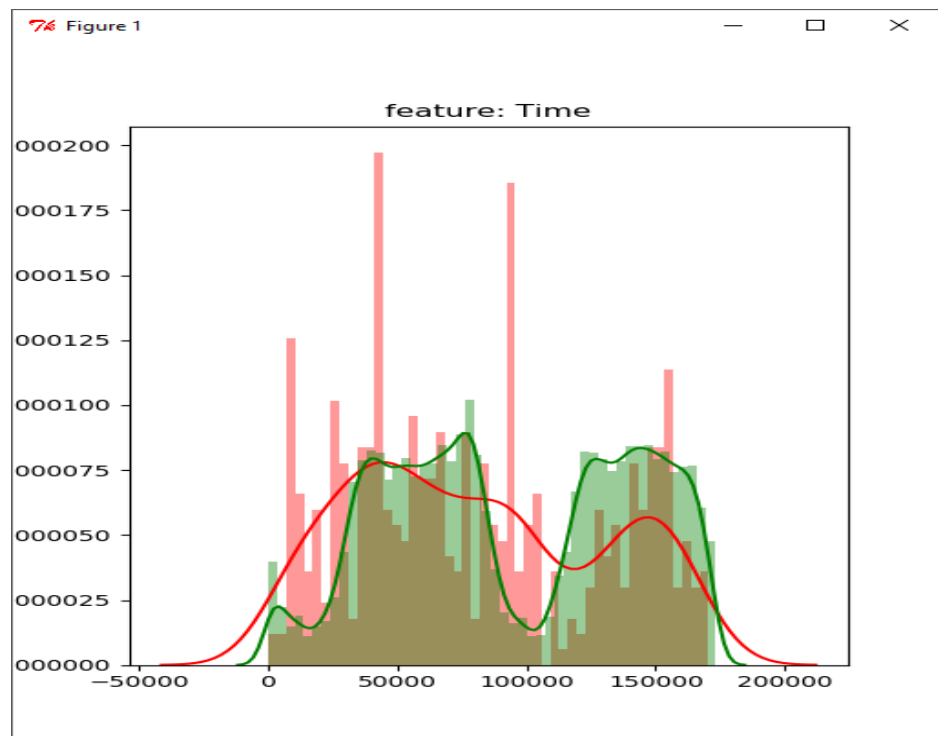


Fig 4.3.2 Correlations of Time

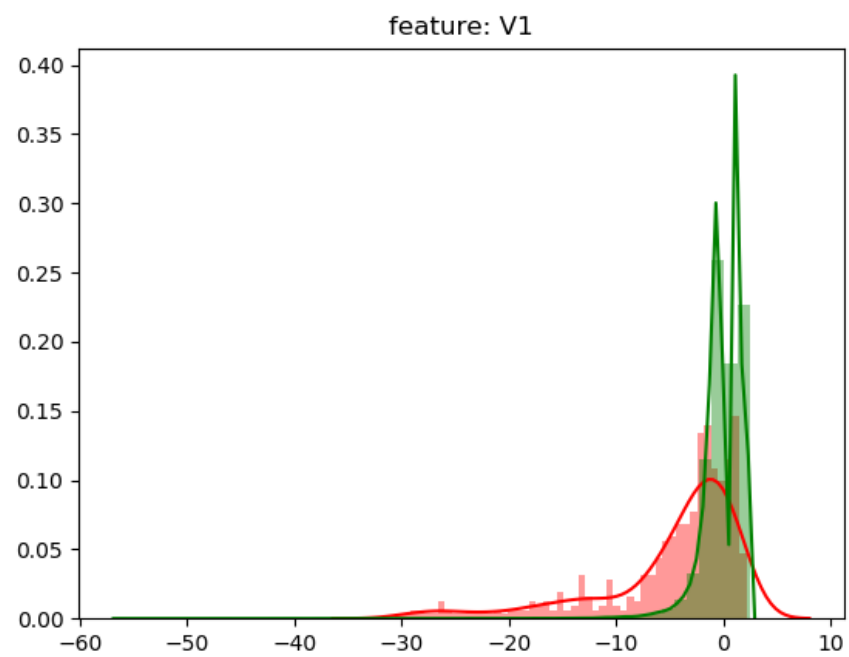


Fig 4.3.3 Correlations of V1

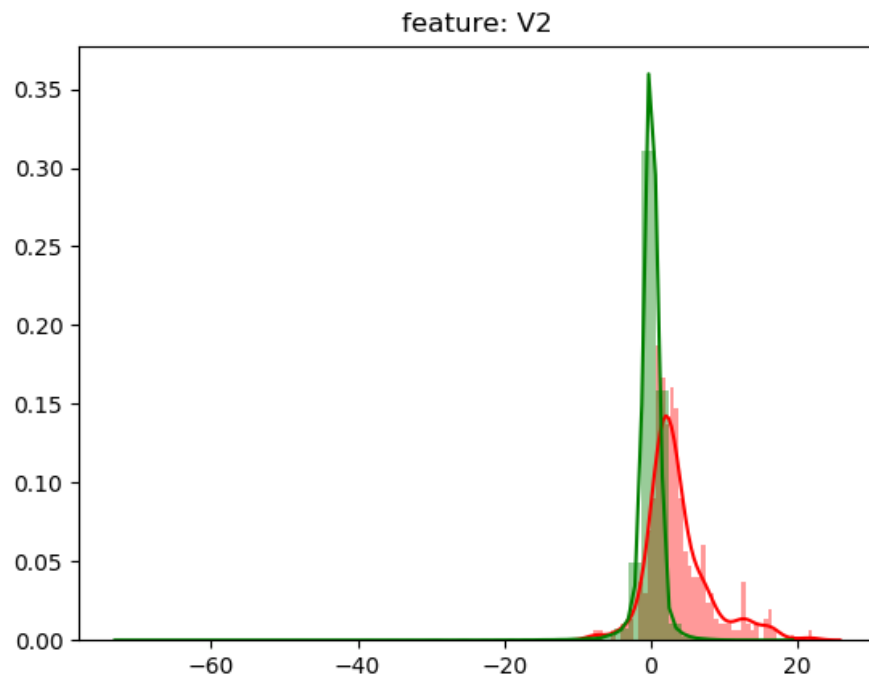


Fig 4.3.4 Correlations of V2

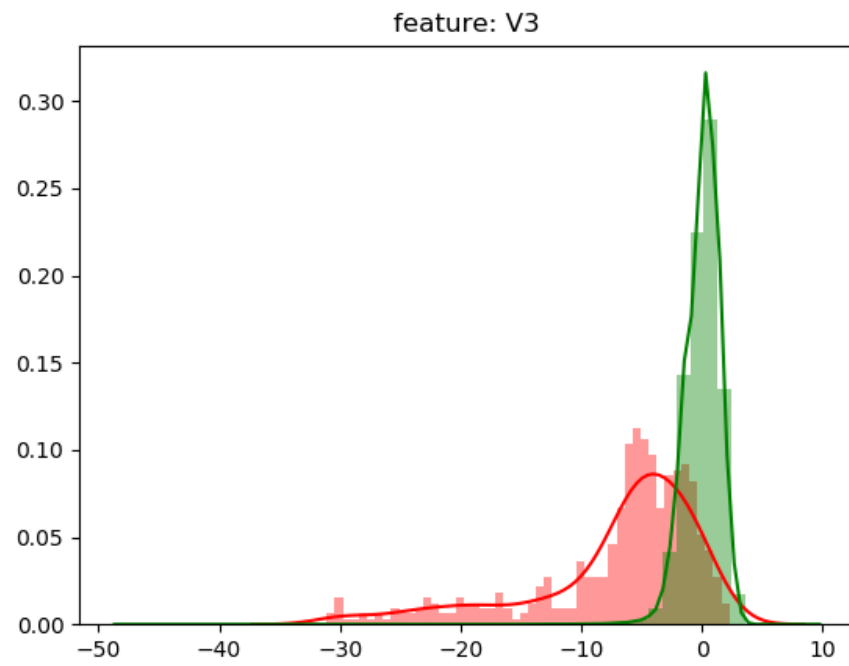


Fig 4.3.5 Correlations of V3

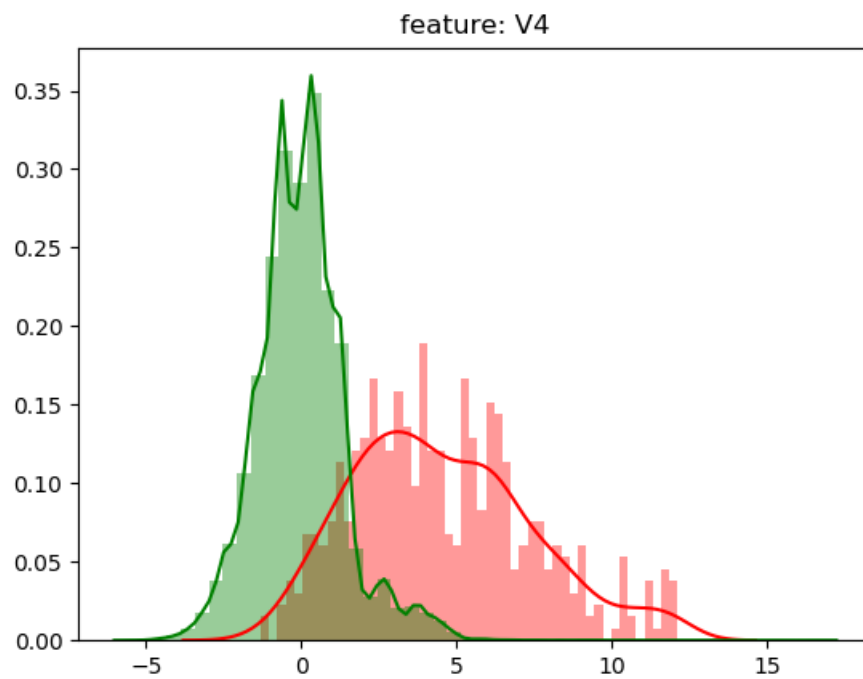


Fig 4.3.6 Correlations of V4

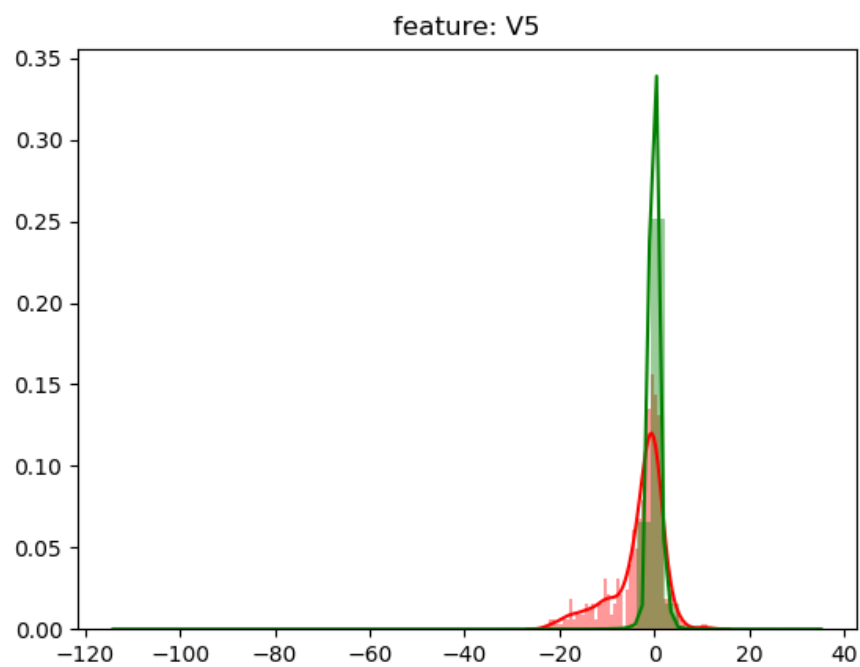


Fig 4.3.7 Correlations of V5



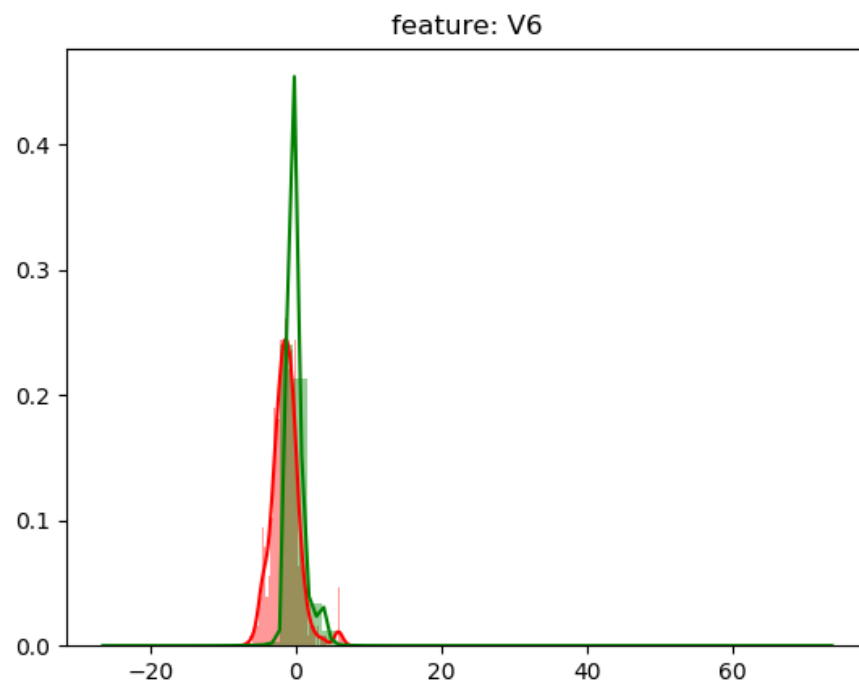


Fig 4.3.8 Correlations of V6

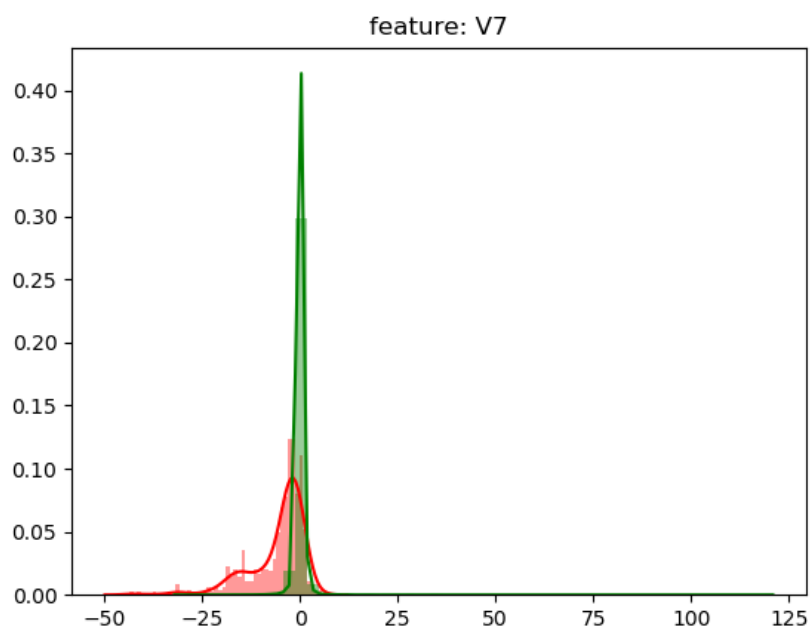


Fig 4.3.9 Correlations of V7

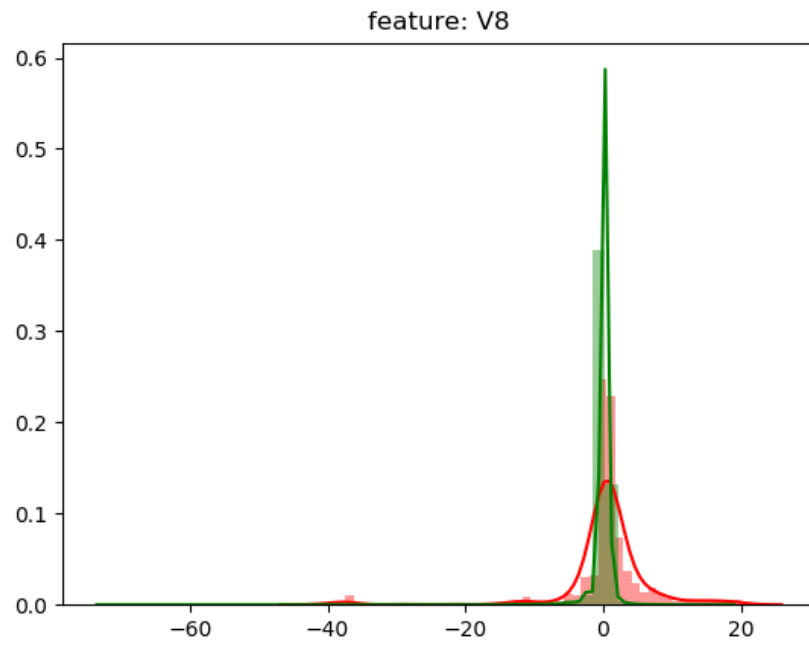


Fig 4.3.10 Correlations of V8

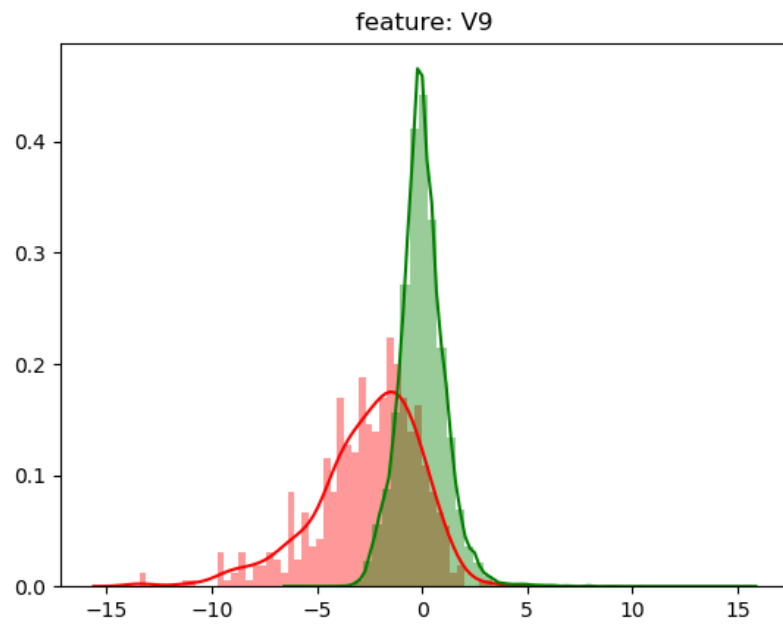


Fig 4.3.11 Correlations of V9

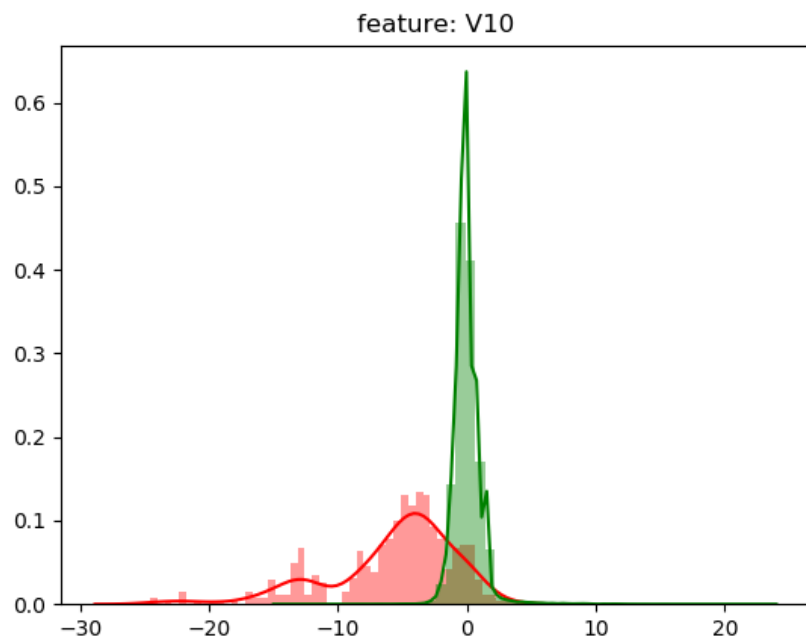


Fig 4.3.12 Correlations of V10

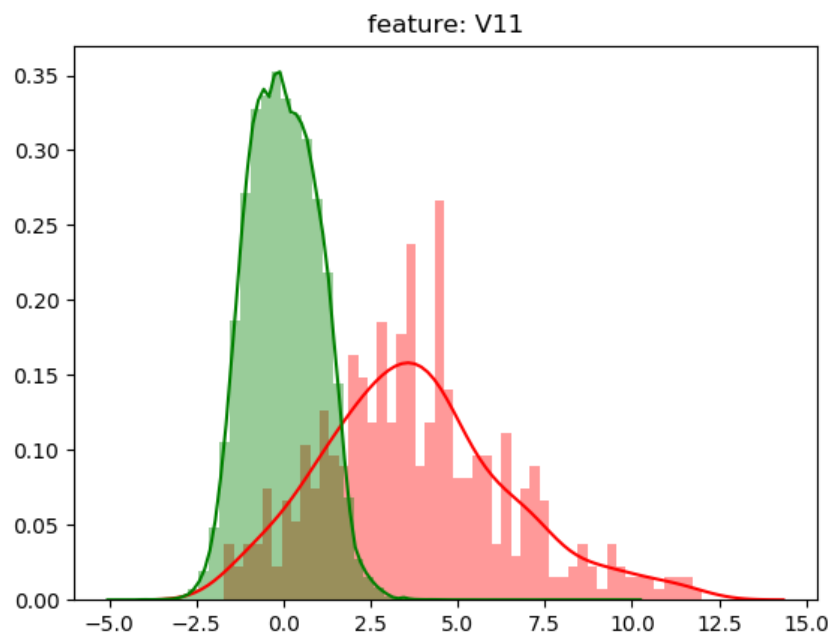


Fig 4.3.13 Correlations of V11

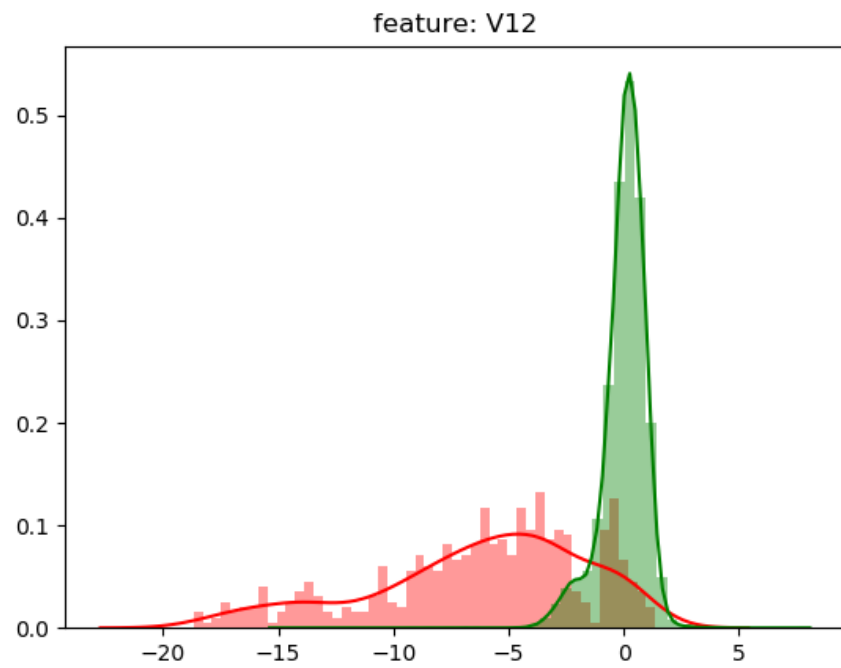


Fig 4.3.14 Correlations of V12

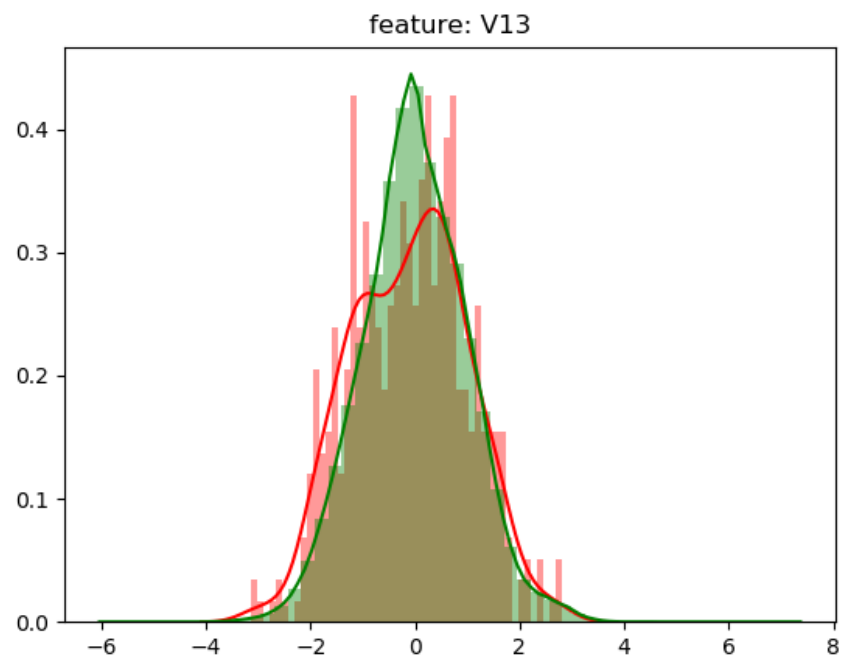


Fig 4.3.15 Correlations of V13

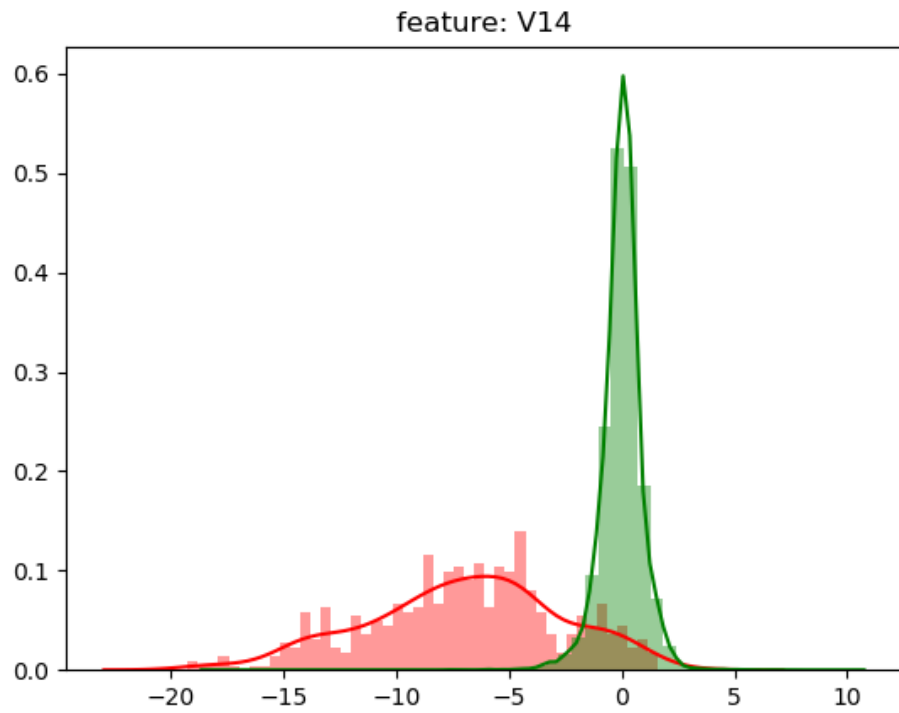


Fig 4.3.16 Correlations of V14

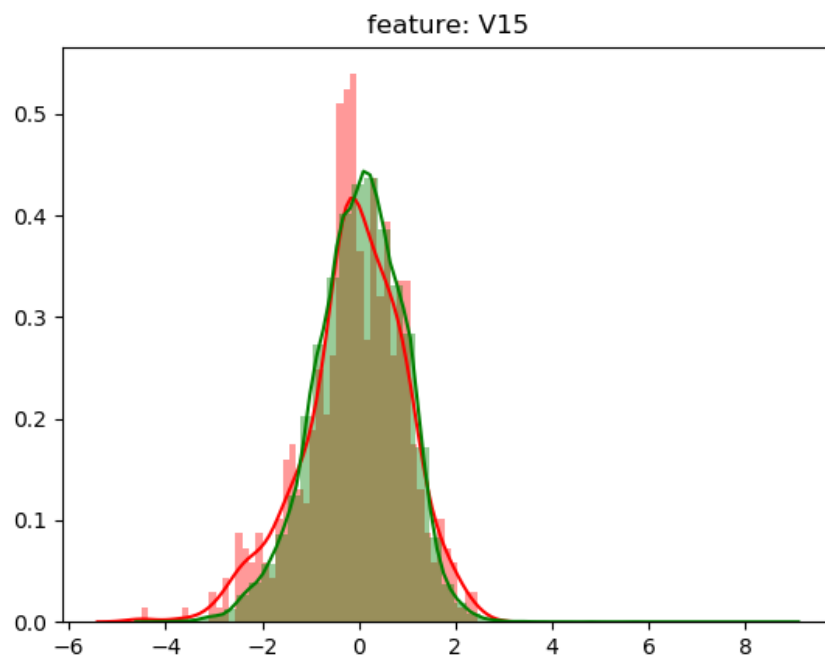


Fig 4.3.17 Correlations of V15

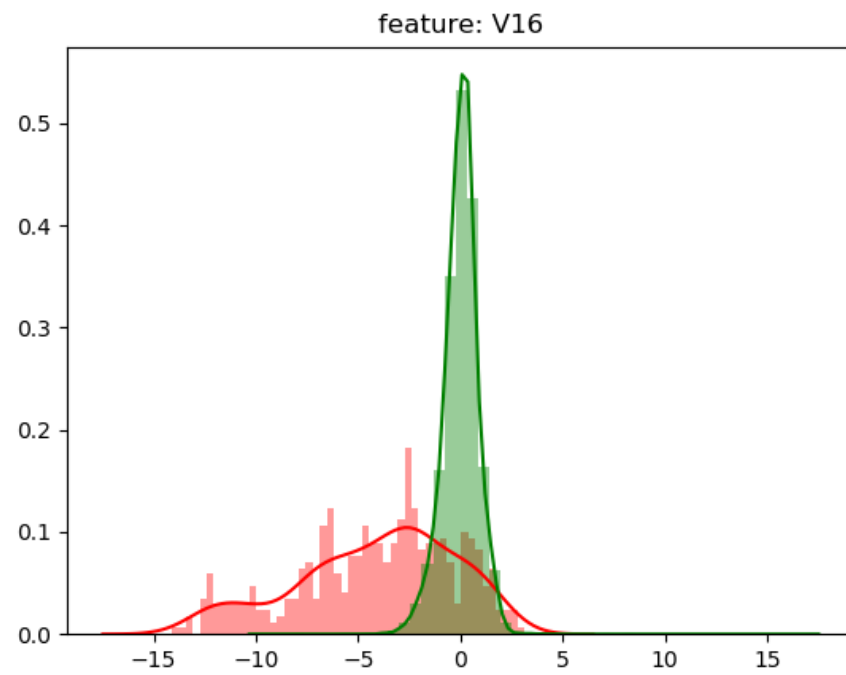


Fig 4.3.18 Correlations of V16

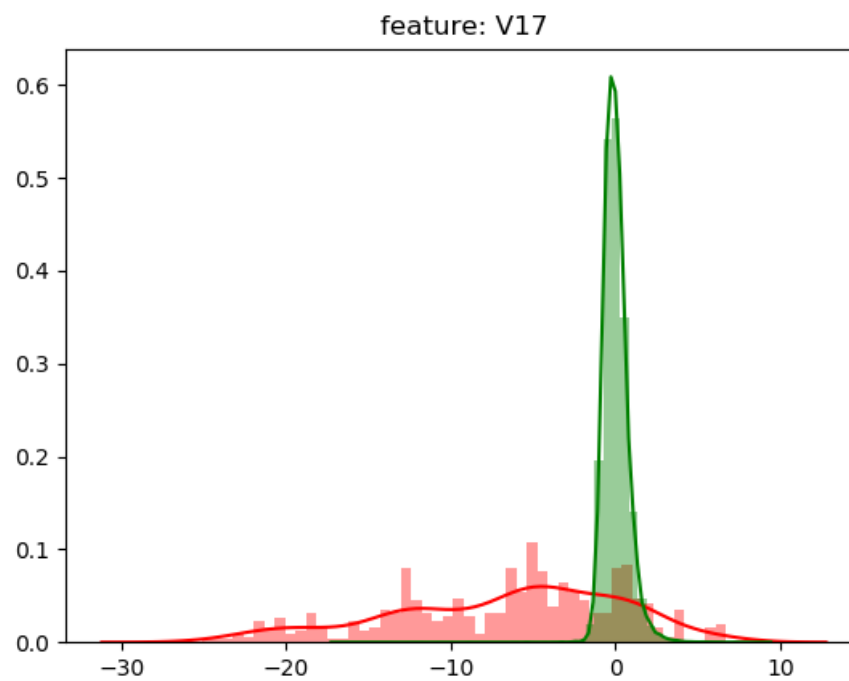


Fig 4.3.19 Correlations of V17

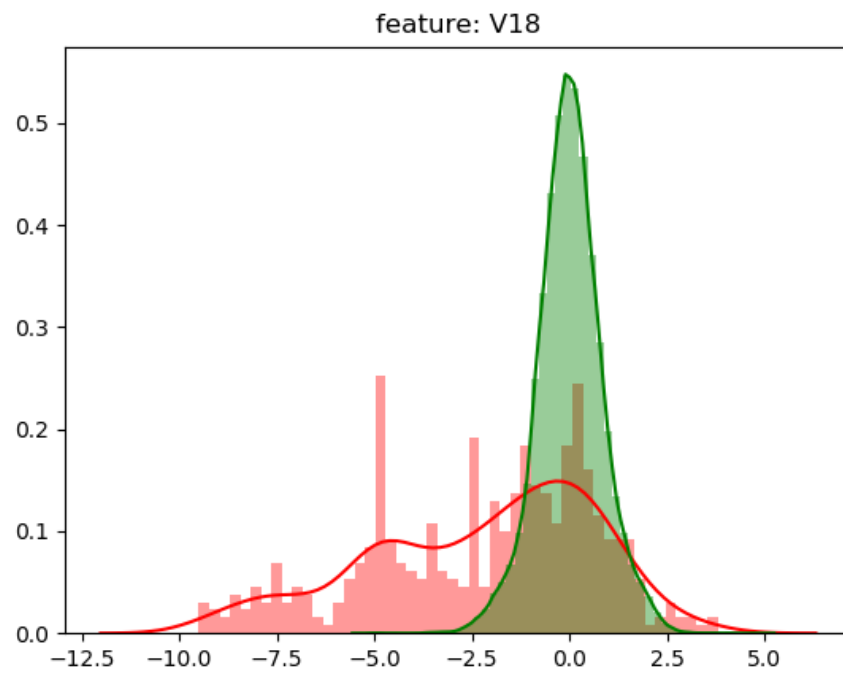


Fig 4.3.20 Correlations of V18

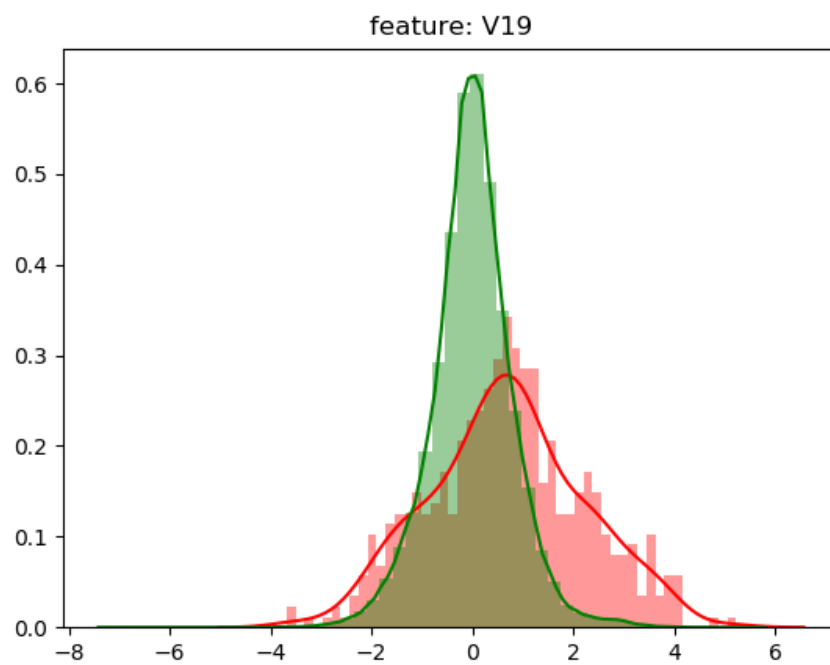


Fig 4.3.21 Correlations of V19

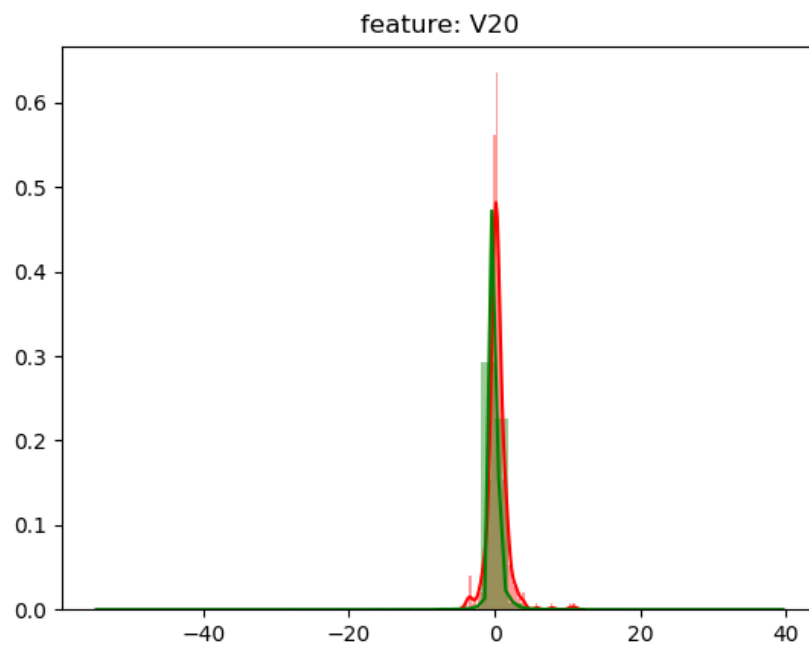


Fig 4.3.22 Correlations of V20

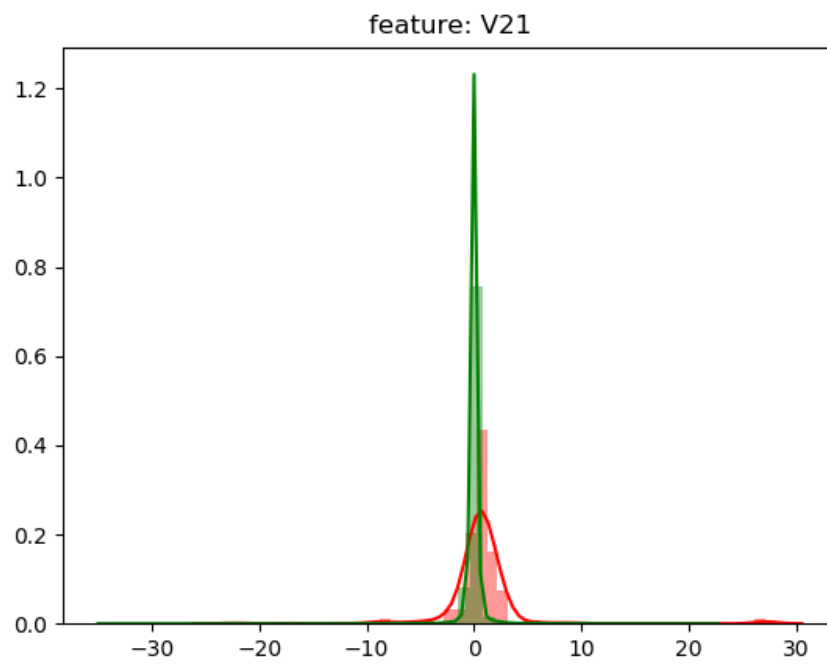


Fig 4.3.23 Correlations of V21



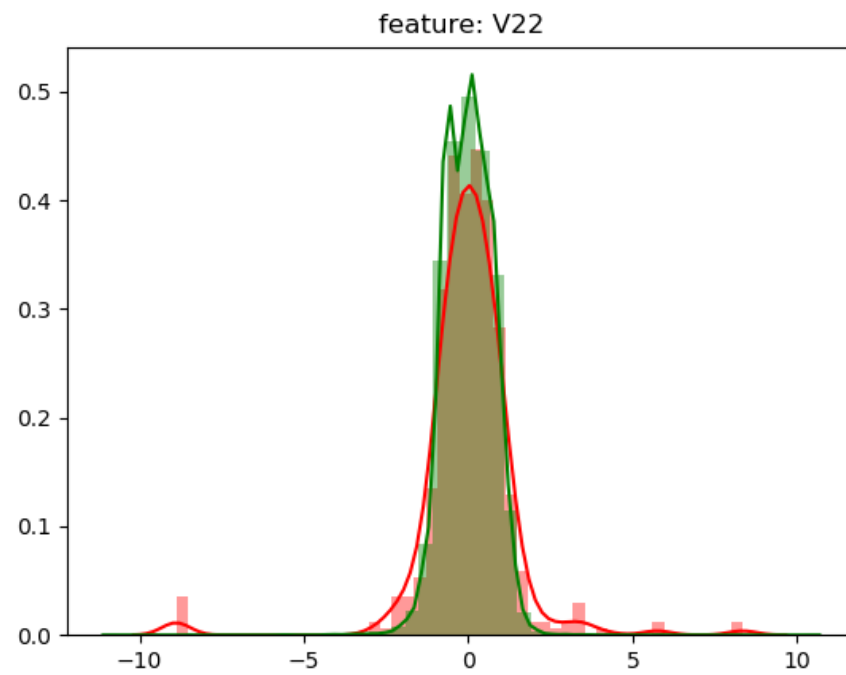


Fig 4.3.24 Correlations of V22

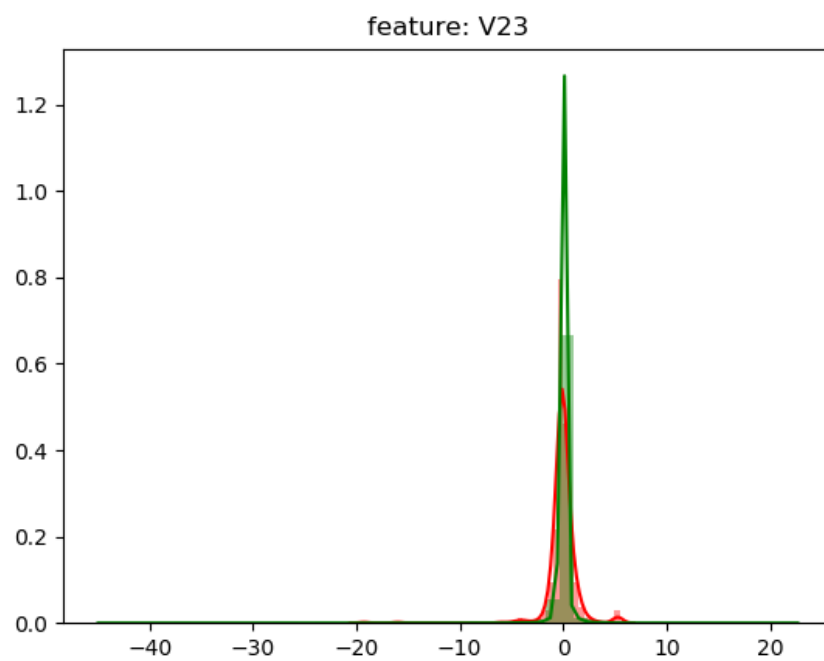


Fig 4.3.25 Correlations of V23

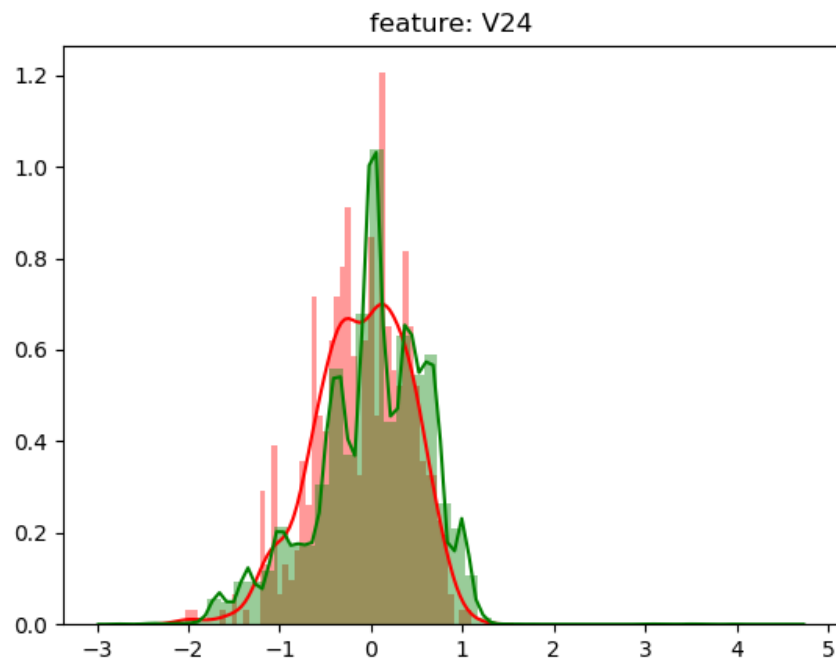


Fig 4.3.26 Correlations of V24

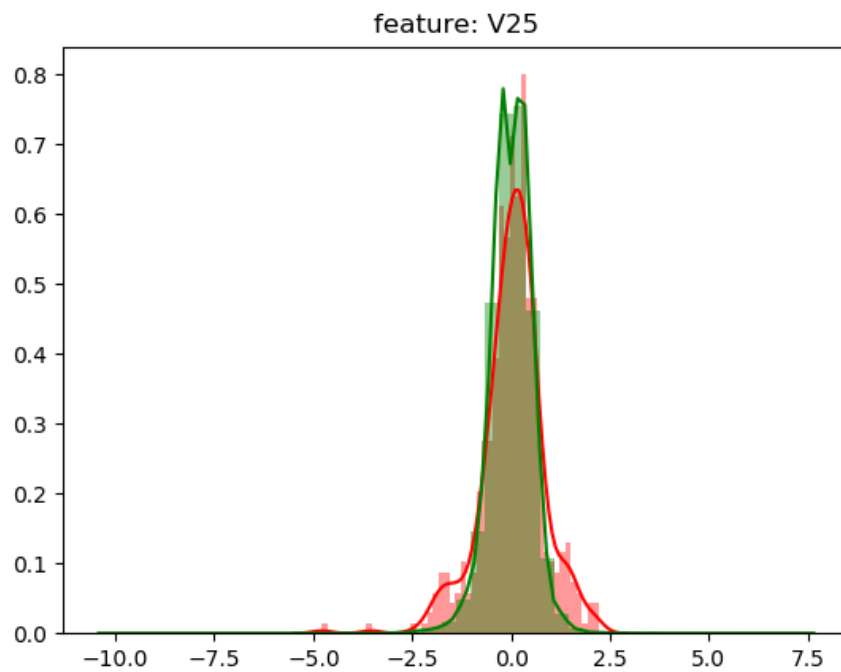


Fig 4.3.27 Correlations of V25

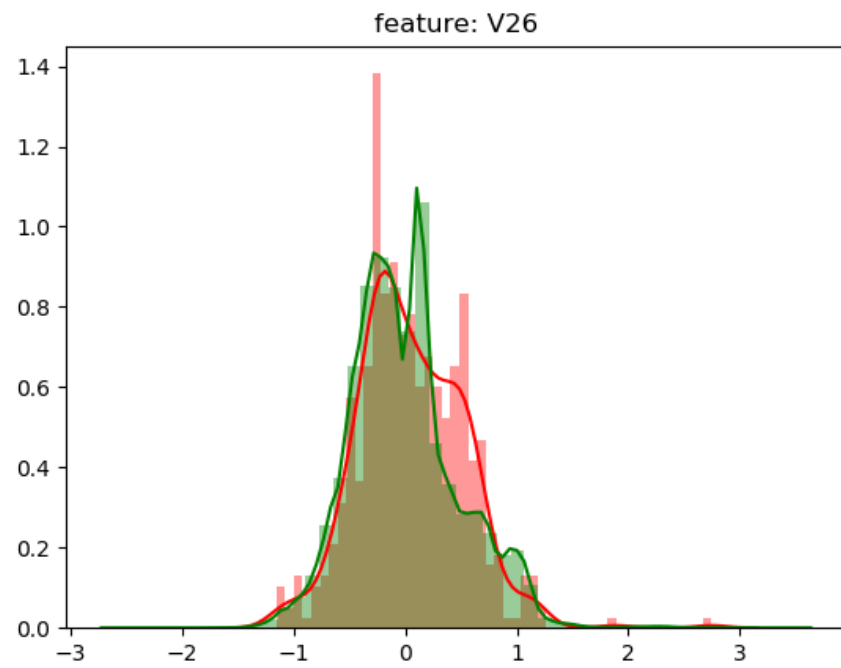


Fig 4.3.28 Correlations of V26

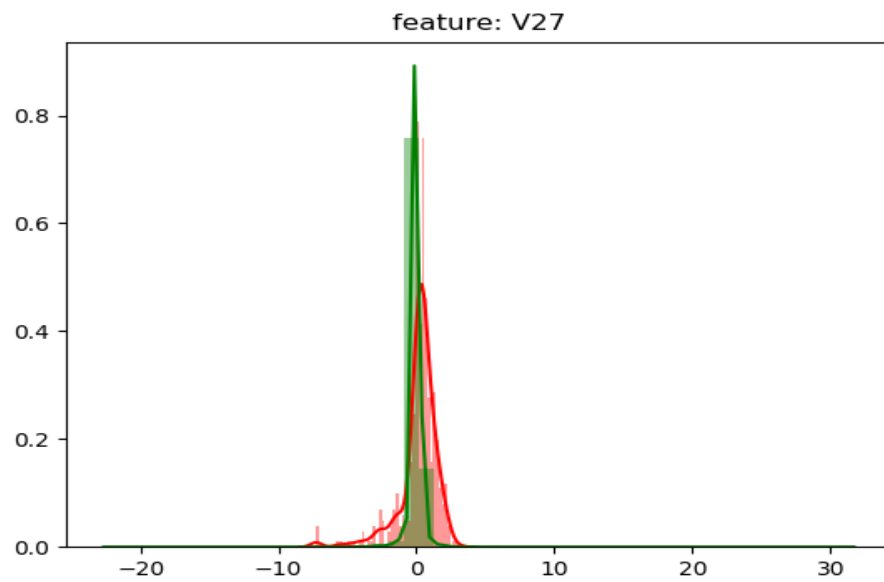


Fig 4.3.29 Correlations of V27

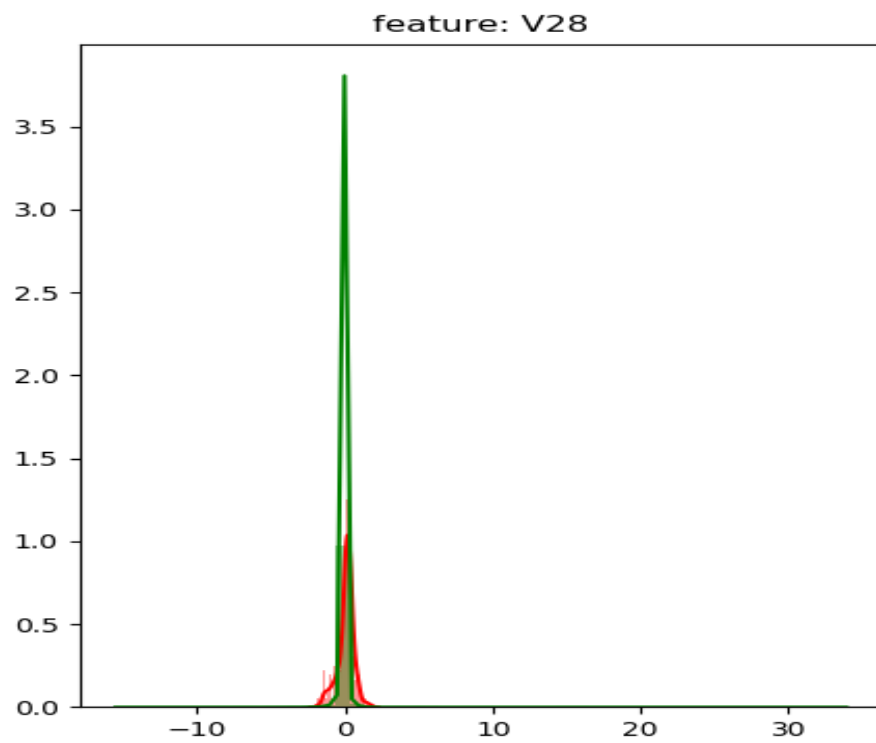


Fig 4.3.30 Correlations of V28

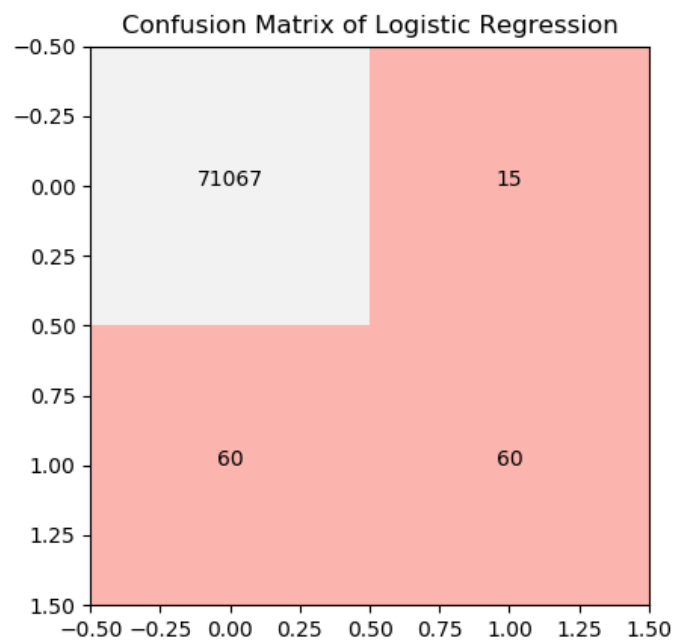


Fig 4.3.31 Confusion Matrix of Logistic Regression

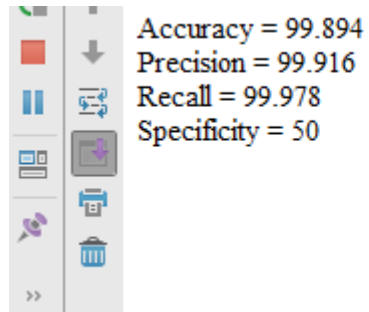


Fig 4.3.32 Metrics of Logistic Regression

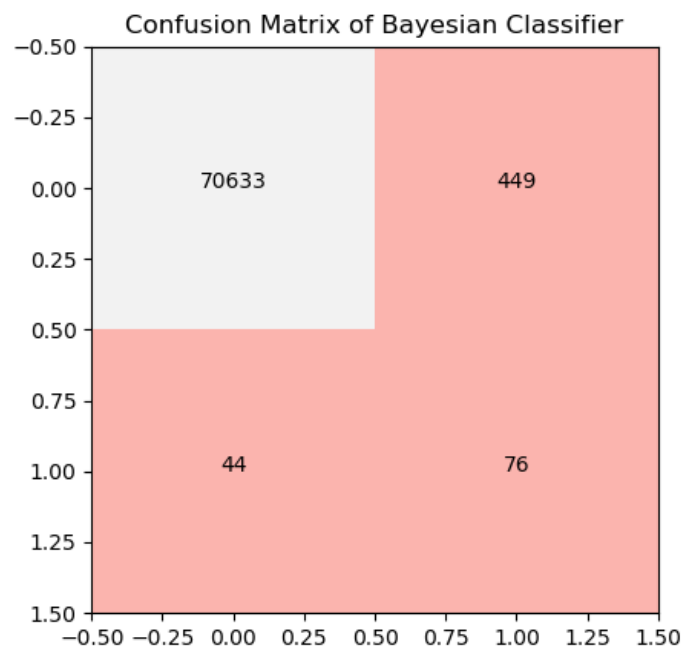


Fig 4.3.33 Confusion Matrix of Bayesian Classifier

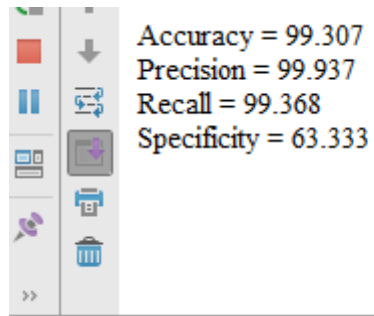


Fig 4.3.34 Metrics of Bayesian Classifier

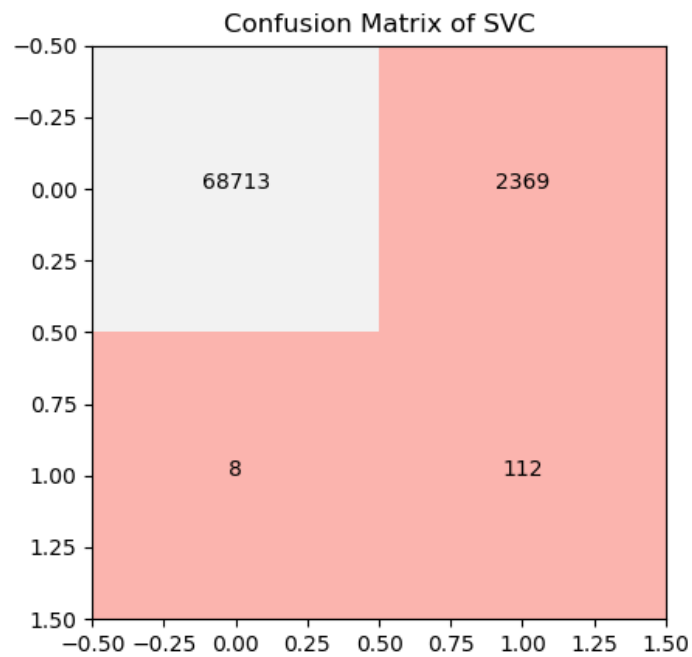


Fig 4.3.35 Confusion Matrix of SVC

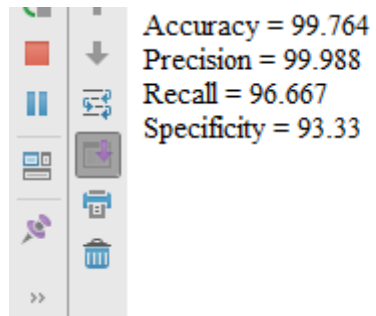


Fig 4.3.36 Metrics of SVC

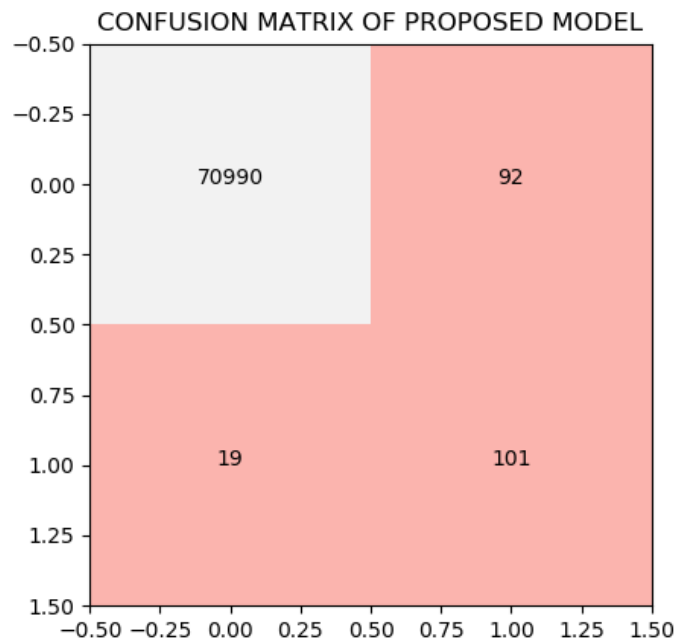
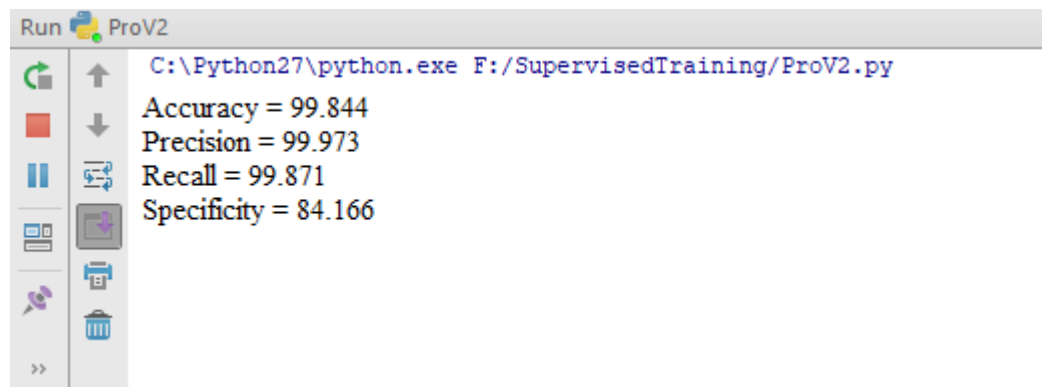


Fig 4.3.37 Confusion Matrix of Proposed CCFDS Model



The screenshot shows a 'Run' window for a program named 'ProV2'. The command line is 'C:\Python27\python.exe F:/SupervisedTraining/ProV2.py'. The output displays four performance metrics: Accuracy = 99.844, Precision = 99.973, Recall = 99.871, and Specificity = 84.166. The window includes a toolbar with icons for running, pausing, and other standard IDE functions.

```
Run ProV2
C:\Python27\python.exe F:/SupervisedTraining/ProV2.py
Accuracy = 99.844
Precision = 99.973
Recall = 99.871
Specificity = 84.166
```

Fig 4.3.38 Metrics of proposed CCFDS Model



## **5. TESTING-TEST DATA SET, TEST CASES**

Software Testing is a critical element of software quality assurance and represents the ultimate service of specification design and coding. The increasing visibility of software as a system element and the attended costs associated with the software failure and motivating forces for well planned, through testing. It is not unusual for a software development to spend between 30 to 40% of total project effort in testing.

System Testing Strategies for this system integrate test case design techniques into a well-planned series of steps that result in the successful construction of this software. It also provides a road map for the developer, the quality assurance organization and the customer, a roadmap that describes the steps to be conducted as path of testing, when these steps are planned and then undertaken and how much effort, time and resources will be required. The test provisions are follows.

### **5.1 Testing Objectives**

The following are the testing objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers as a yet undiscovered error.

The above objectives imply a dramatic change in view point. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically different clauses of errors and do so with minimum amount of time and effort.

If testing is conducted successfully, it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification and that performance requirements appear to have been met. In addition, data collected as testing is conducted provides a good indication of software. Testing can't show the absence of defects, it can only show that software errors are present. It is important to keep this stated in mind as testing is being conducted.

## 5.2 Testing Principles

Before applying methods to design effective test cases, a software engineer must understand the basic principles that guide software testing.

- All tests should be traceable to customer requirements.
- Tests should be planned before testing begins.
- Testing should begin “in the small” and progress towards testing “in the large”.
- Exhaustive testing is not possible.

## 5.3 Testing Strategies

A strategy for software testing indicates software test case design methods in to a well-planned series of steps that results in the successful construction of software. The strategy provides a road map that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time and resources will be required. It must be rigid enough to promote reasonable planning and management tracking as the project progresses. The strategies for testing are envisioned by the following methods. A number of software testing strategies have been proposed. All provide the software developer with a template for testing and all following generic characteristics.

- To perform effective testing. A software team should conduct effective formal technical reviews. By doing this, many errors will be eliminated before testing commences.
- Testing begins at the component level and works “outward” towards the integration of entire computer based system.
- Different testing techniques are appropriate at different points in time.
- Testing is conducted by the developer of the software and an independent test group.
- Testing and debugging are different activities, but debugging must accommodate in any testing strategy.

A strategy for software testing must accommodate low level test that are necessary to verify that a small source code segment has been correctly implemented as well as high

level tests that validates major system functions against customer requirements. A strategy must provide guidance for the practitioner and set of milestones for the manager.

## **5.4 Types of Testing**

The primary objective of test case design is to derive a set of tests that have the highest likelihood for uncovering errors in the software. To accomplish this objective two different categories of test case design techniques are used.

### **1. White-Box Testing**

White box testing sometimes called glass box testing is a test case designs that focus on the program control structure. Test cases are derived to ensure that

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical design on their true and false sides.
3. Executes all loops at their boundaries and within their operational boundaries.
4. Exercise internal data structure to ensure their validity.

Several methods are used in the white box testing.

### **2. Black Box Testing**

Tests can be conducted at software interface by knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational, at the same time searching for errors is called black box testing, sometimes called as behavioural testing. Black box testing is not an alternative to white box techniques. Rather it is a complementary approach that is likely to uncover a different class of errors than white box methods. Black box tests are designed to uncover errors in functional requirements without regard to the internal workings of a program. Black box testing techniques focus on the information domain of the software.

Black box testing attempts to find errors in the following categories.

1. Incorrect or missing functions.
2. Interface errors.
3. Errors in the data structures or external database access.
4. Performance errors.

5. Initialization and termination errors.

### **3. System Testing**

System tests are designed to validate a fully developed system with a view to assuming that it meets its requirements. System testing is actually a series of different tests, whose primary purpose is to be fully exercising the computer-based system. In this system, although each test has a different purpose, all the works are verified to ensure that all system elements have been properly integrated and performed allocated functions.

There are three kinds of system testing:

1. Alpha Testing: Alpha testing refers to the system testing that is carried by the customer within the organization along with the developer. The alpha test are conducted in controlled manner.
2. Beta Testing: Beta testing is the system performed by a selected group of customers, the developer is not present at the site and the user will inform the problems that are encountered during testing. The software developer makes the necessary changes and submits to the customer.
3. Acceptance Testing: Acceptance testing is the system testing performed by the customer to whether or not to accept the delivery of the system.

### **4. Unit Testing**

Unit testing focuses verification effort on the smallest unit of the software design, the module. Using the detailed design description as a guide, important control paths are tested to uncover errors with the boundary of the module for the following modules. All the statements in the module are executed at least once. From this we can ensure that all independent paths through the control structures are exercised.

### **5. Integration Testing**

Integration testing is a systematic technique for constructing the program structures and to conduct tests for uncovered errors with interfacing.

## 5.5 TESTCASES

S.No	Input	Expected Output	Obtained Output	Remarks
1	Training Transactions data to Logistic Regression	Clusters of Fraud and Genuine Transactions	Clusters of Fraud and Genuine Transactions	Pass
2	Training Transactions data to Bayesian Classifier	Clusters of Fraud and Genuine Transactions	Clusters of Fraud and Genuine Transactions	Pass
3	Training Transactions data to SVC	Clusters of Fraud and Genuine Transactions	Clusters of Fraud and Genuine Transactions	Pass
4	User Transaction to Logistic Regression	Probability of the transaction to fall in genuine cluster	Probability of the transaction to fall in genuine cluster	Pass
5	User Transaction to Bayesian Classifier	Probability of the transaction to fall in genuine cluster	Probability of the transaction to fall in genuine cluster	Pass
6	User Transaction to SVC	Probability of the transaction to fall in genuine cluster	Probability of the transaction to fall in genuine cluster	Pass
7	Probabilities of transaction from 3 algorithms	Merged Fraud Score	Merged Fraud Score	Pass
8	Merge score to Threshold Verifier to check limits (within limits)	Fraud or Genuine Identification	Fraud or Genuine Identified	Pass
9	Merge score to Threshold Verifier to check limits (outside limits)	Transaction to be passed to Decision Tree Classifier	Transaction passed to Decision Tree Classifier	Pass
10.	Transaction to Decision Tree Classifier	Fraud or Genuine Identification	Fraud or Genuine Identified	Pass

## 6. IMPLEMENTATION-REQUIREMENTS, INSTALLATION PROCEDURE

### 6.1 Requirements

#### PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

#### Libraries Used:

1. **NumPy** stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++
2. **SciPy** stands for Scientific Python. SciPy is built on NumPy. It is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.
3. **Matplotlib** for plotting vast variety of graphs, starting from histograms to line plots to heat plots.. You can use Pylab feature in ipython notebook (ipython notebook – pylab = inline) to use these plotting features inline. If you ignore the inline option, then pylab converts ipython environment to an environment, very similar to Matlab. You can also use Latex commands to add math to your plot.
4. **Pandas** for structured data operations and manipulations. It is extensively used for data munging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community.

5. **Scikit Learn** for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
6. **Seaborn** for statistical data visualization. Seaborn is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.

## **HADOOP**

Apache Hadoop is an open-source software framework used for distributed storage and processing of datasets of big data using the Map-Reduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a Map-Reduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data they have access to.

## **JAVA**

Java is a network-friendly programming language invented by Sun Microsystems. Java is a complete software ecosystem that represents different values to different types of users. Java technology is the first software technology that simply works without a struggle. Users are delighted to see applications run reliably and compatibly on such an incredible array of network products - from PCs, game players, and mobile phones to home appliances and automotive electronics.

Java technology is an object-oriented, platform-independent, multithreaded programming environment. It is the foundation for Web and networked services, applications, platform independent desktops, robotics, and other embedded devices.

## 6.2 INSTALLATION PROCEDURES

### 1. Python

- Open terminal (Ctrl+Alt+T) and run the commands:  
`sudo apt update`  
`sudo apt upgrade`  
`sudo apt install python2.7 python-pip`
- To install packages using pip use the command:  
`pip2 install <package-name>`

### 2. Java

- Open terminal (Ctrl+Alt+T) and run the command:

```
sudo add-apt-repository ppa:webupd8team/java
```

Type in your password when it asks and hit Enter.

- Update and install the installer script:

Run commands to update system package index and install Java installer script:

```
sudo apt update; sudo apt install oracle-java8-installer
```

You may replace oracle-java8-installer with oracle-java9-installer to install Java 9. While the install process, you have to accept Java license to continue downloading & installing Java binaries.

- Check the Java version

To check the Java version after installing the package, run command:

```
javac -version
```

- Set Java environment variables

The PPA also contains a package to automatically set Java environment variables, just run command:

```
sudo apt install oracle-java8-set-default
```

### 3. HDFS

- `$ sudo apt-get update`
- `$ sudo apt-get install default-jdk`
- `$ java -version`
- `$ sudo apt-get install ssh`



- \$ sudo apt-get install rsync
- \$ ssh-keygen -t dsa -P '' -f ~/.ssh/id\_dsa
- \$ cat ~/.ssh/id\_dsa.pub >> ~/.ssh/authorized\_keys
- \$ wget -c http://mirror.olnevhost.net/pub/apache/hadoop/common/current/hadoop-2.6.0.tar.gz
- \$ sudo tar -zxvf hadoop-2.6.0.tar.gz
- \$ sudo mv hadoop-2.6.0 /usr/local/Hadoop
- \$ update-alternatives --config java
- \$ sudo gedit ~/.bashrc

#Hadoop Variables

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

export HADOOP\_HOME=/usr/local/hadoop

export PATH=\$PATH:\$HADOOP\_HOME/bin

export PATH=\$PATH:\$HADOOP\_HOME/sbin

export HADOOP\_MAPRED\_HOME=\$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=\$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=\$HADOOP\_HOME

export YARN\_HOME=\$HADOOP\_HOME

export

HADOOP\_COMMON\_LIB\_NATIVE\_DIR=\$HADOOP\_HOME/lib/native

export HADOOP\_OPTS="-Djava.library.path=\$HADOOP\_HOME/lib"

- \$ source ~/.bashrc
- \$ cd /usr/local/hadoop/etc/hadoop
- \$ sudo gedit hadoop-env.sh

#The java implementation to use.

export JAVA\_HOME="/usr/lib/jvm/java-7-openjdk-amd64"

- \$ sudo gedit core-site.xml

<configuration>

<property>

```

        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>

```

- \$ sudo gedit yarn-site.xml

```

<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
    <value> org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>

```

- \$ sudo cp mapred.site.xml.template mapred-site.xml
- \$ sudo gedit mapred-site.xml

```

<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>

```

- \$ sudo gedit hdfs-site.xml

```

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>

```

```
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop/hadoop_data/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop/hadoop_store/hdfs/datanode</value>
</property>
</configuration>
```

- \$ cd
- \$ mkdir -p /usr/local/hadoop/hadoop\_data/hdfs/namenode
- \$ mkdir -p /usr/local/hadoop/hadoop\_data/hdfs/datanode
- \$ sudo chown chaalpritam:chaalpritam -R /usr/local/Hadoop
- \$ hdfs namenode -format
- \$ start-all.sh
- \$ jps

## **7. CONCLUSION**

Online fraud detection is challenging due to the burst amount of trading transactions that are happening every day. Proposed model aims at fusing different detection algorithms to improve accuracy. Big Data technologies, which help to build a scalable, fault-tolerant and high performance system, are used. A series of anti-fraud strategies can be adopted to prevent banks from great losses before and reduce risks after fraud detection is done using the proposed model. It can also be applied in other similar application fields, for example internet advertising fraud detection, telecom fraud detection and so on.

## **8. BIBLIOGRAPHY**

### **8.1 List of Book References**

- Online Credit Card Fraud Detection: A Hybrid Framework with Big Data Technologies , by You Dai, Jin Yan, Xiaoxin Tang, Han Zhao and Minyi Guo
- Hands on Machine Learning with Scikit-Learn and TensorFlow by Aurelien Geron
- Hadoop: The Definitive Guide, 4th Edition

### **8.2 List of Web References**

- <http://scikit-learn.org/stable/documentation.html>
- <https://matplotlib.org/contents.html>
- <https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/learning-path-data-science-python/>
- <https://community.hortonworks.com/articles/92321/interacting-with-hadoop-hdfs-using-python-codes.html>