

1. INTRODUCTION

Nowadays the usage of credit cards has dramatically increased. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising. In this paper, we model the sequence of operations in credit card transaction processing using a Random Forest to show how it can be used for the detection of frauds. In both algorithms is initially trained with the normal behaviour of a cardholder. If an incoming credit card transaction is not accepted by the trained with sufficiently high probability, it is considered to be fraudulent. At the same time, we try to ensure that genuine transactions. We present detailed experimental results to show the effectiveness of our approach and compare it with other techniques.

1.1 Objective

Billions of dollars of loss are caused every year by the fraudulent credit card transactions. Fraud is old as humanity itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there is definitely a need to solve the problem of credit card fraud detection. Moreover, the development of new technologies provides additional ways in which criminals may commit fraud. The use of credit cards is prevalent in modern day society and credit card fraud has been kept on growing in recent years. Huge Financial losses has been fraudulent affects not only merchants and banks, but also individual person who are using the credits. Fraud may also affect the reputation and image of a merchant causing non-financial losses that, though difficult to quantify in the short term, may become visible in the long period. For example, if a cardholder is victim of fraud with a certain company, he may no longer trust their business and choose a competitor.

1.2 Methodology

There are various fraudulent activities detection techniques has implemented in credit card transactions have been kept in researcher minds to methods to develop models based on artificial intelligence , data mining, fuzzy logic and machine learning. Credit card fraud detection is an extremely difficult, but also popular problem to solve. In our

proposed system we built the credit card fraud detection using Machine learning. With the advancement of machine learning techniques. Machine learning has been recognized as a successful measure for fraud detection. A great deal of data is transferred during online transaction processes, resulting in a binary result: genuine or fraudulent. Online businesses are able to identify fraudulent transactions accurately because they receive chargebacks on them. Within the sample fraudulent datasets, features are constructed. These are data points such as the age and value of the customer account, as well as the origin of the credit card. There can be hundreds of features and each contributes, to varying extents, towards the fraud probability. Note, the degree in which each feature contributes to the fraud score is not determined by a fraud analyst, but is generated by the artificial intelligence of the machine which is driven by the training set. So, in regards to the card fraud, if the use of cards to commit fraud is proven to be high, the fraud weighting of a transaction that uses a credit card will be equally so. However, if this were to diminish, the contribution level would parallel. Simply put, these models self-learn without explicit programming such as with manual review. Credit card fraud detection using Machine learning is done by deploying the classification and regression algorithms. We use supervised learning algorithm such as Decision tree algorithm to classify the fraud card transaction in online or by offline. Random forest has better efficiency and accuracy than the other machine learning algorithms. Random forest aims to reduce the previously mentioned correlation issue by choosing only a subsample of the feature space at each split. Essentially, it aims to make the trees de-correlated and prune the trees by setting a stopping criteria for node splits, which I will cover in more detail later.

1.2.1 Dataset

Data used in this paper is a set of product reviews collected from credit card transactions records. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called *labelled data*.

```
In [6]: data.head(20)
```

```
Out[6]:
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
0	51435.0	1.197490	-0.352125	-0.135904	0.222100	0.231128	1.086617	-0.420363	0.391464	0.672499	...	-0.337999	-0.963921	-0.121931	-1.723430
1	78049.0	0.976047	-0.289947	1.465321	1.300002	-1.382887	-0.479586	-0.632572	0.064533	0.710743	...	0.322829	0.790185	-0.101364	0.730461
2	157168.0	-1.395302	0.478266	-0.584911	-1.201527	0.928544	-0.743618	0.755504	-0.141397	-2.118499	...	0.282803	0.903183	-0.444694	0.696438
3	69297.0	1.278014	-0.672705	-0.425494	-0.777398	-0.582088	-0.880396	-0.103505	-0.203036	-1.241653	...	0.256003	0.408211	-0.274815	0.028707
4	144504.0	-0.312745	-1.202565	2.249806	-0.297210	-0.963389	1.207532	-0.837776	-0.057654	1.121421	...	-0.274386	0.682305	0.432717	0.722364
5	43843.0	-0.152063	-3.309440	-0.875095	-0.354323	-1.281074	0.575960	0.241441	-0.075757	-0.503415	...	0.415543	-0.819220	-0.759940	-1.322200
6	124531.0	-0.884906	-0.408735	1.796732	-0.637739	-0.565369	1.693424	0.287936	0.623672	0.892499	...	0.528267	1.361604	0.397627	-1.068664
7	147090.0	-6.517591	3.610050	-4.894036	-0.419986	-3.826823	-1.202157	-2.158948	4.173289	-0.005825	...	-0.158756	-0.941744	0.192155	0.002764
8	125405.0	0.573327	0.557713	-0.718471	-0.833950	1.017760	-0.275706	0.752081	-0.021168	-0.053536	...	0.366566	1.094066	-0.118825	0.235184
9	23413.0	1.197664	0.184014	0.722613	0.776004	-0.601469	-0.776805	-0.238624	-0.057559	1.310535	...	-0.359904	-0.920248	0.177993	0.423467
10	58009.0	1.360571	-0.815752	0.253770	-1.189123	-0.364303	1.087002	-1.143574	0.271348	-0.690392	...	0.373865	1.029942	-0.333073	-1.627205
11	19757.0	-0.403430	1.087224	1.153807	-0.170787	0.401999	-0.758542	0.885464	-0.167590	1.048641	...	-0.062636	0.372130	-0.059322	0.401879
12	120507.0	0.013174	1.064257	-2.258120	-0.006158	0.647395	-1.568600	1.962088	-0.288213	-0.629780	...	0.648958	1.982292	0.239520	0.071280
13	123338.0	-0.319508	-1.690567	0.269720	-2.386524	0.988088	-0.420967	-0.738794	-0.126914	-1.973004	...	-0.036959	0.203625	0.020597	-1.410946
14	118956.0	-0.179003	0.309362	-0.112848	-0.092127	-0.500665	-0.031737	-0.492223	0.469185	-1.705332	...	0.309462	0.734239	-0.028024	-0.617519
15	125257.0	-4.534103	-11.785517	-6.736955	1.651770	-2.394427	1.347170	4.767352	-1.032555	-1.310538	...	2.630959	-0.234572	-3.216360	-0.308345
16	131142.0	-0.503272	1.037077	1.336844	-0.061138	0.557151	-0.678385	1.029498	-0.220941	-0.844560	...	-0.260920	-0.688691	-0.284787	0.028036
17	69190.0	-0.320096	0.946312	1.391710	0.053293	0.065174	-0.865741	1.095034	-0.323093	-0.308008	...	-0.285553	-0.644304	0.050585	0.330817
18	36346.0	-0.634011	0.555985	0.829155	-1.747728	0.279131	0.792218	-0.826181	-2.333646	-0.365675	...	-1.621988	-0.976576	-0.180052	-1.199925
19	38316.0	1.206824	-0.051713	0.255879	-0.097189	-0.396809	-0.709528	0.013892	-0.123532	0.184213	...	-0.186596	-0.556043	0.116790	0.132836

20 rows x 31 columns

1.2.2 Introduction

Fraud detection involves monitoring the activities of populations of users in order to estimate, perceive or avoid objectionable behaviour, which consist of fraud, intrusion, and

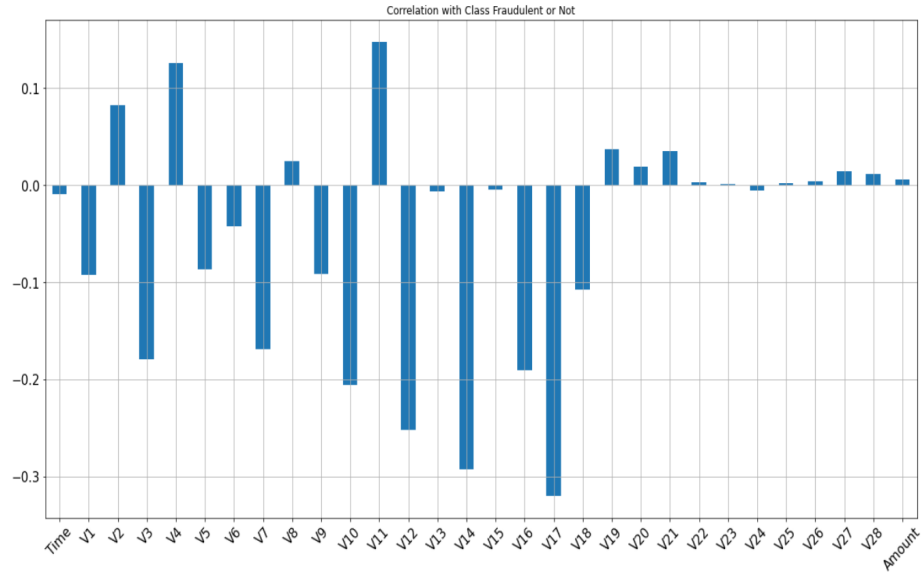
defaulting. This is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated. This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of time.

These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize. Machine learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent. The investigators provide a feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time.

1.2.3 Data Pre-Processing

Organize your selected data by formatting, cleaning and sampling from it. Three common data pre-processing steps are:

- **Formatting:** The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.
- **Cleaning:** Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.
- **Sampling:** There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset. This can be done by finding out the correlation between the attributes.



1.2.4 Feature Extraction

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data.

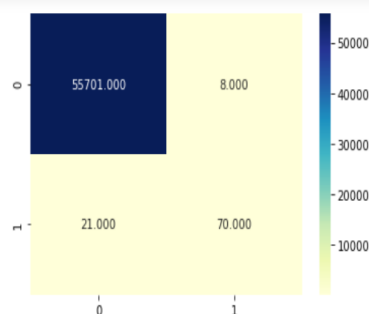
1.2.5 Evaluation Model

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over-fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance. Performance of each classification model is estimated base on its averaged.

The result will be in the visualized form. Representation of classified data in the form of graphs. **Accuracy** is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

Random Forest:

- Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity.
- Random forest is a supervised learning algorithm. The “forest” it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general method of bagging is that a combination of learning models increases the overall result.
- One big advantage of random forest is that it can be useful for both classification and regression problems, which form the majority of current machine learning systems.
- Random forests are generally used as “blackbox” models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configurations.



```
In [23]: from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, plot_roc_curve
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
In [24]: results_testset = pd.DataFrame([[ 'RandomForest', acc, 1-rec, rec, prec, f1]],
columns = ['Model', 'Accuracy', 'FalseNegRate', 'Recall', 'Precision', 'F1 Score'])
results_testset
```

```
Out[24]:
```

	Model	Accuracy	FalseNegRate	Recall	Precision	F1 Score
0	RandomForest	0.99948	0.230769	0.769231	0.897436	0.828402

Decision tree algorithm:

- Decision tree learning or induction of decision trees is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item (represented in branches) to conclusions about the item's target value (represented in the leaves).
- Tree models where the target variable can take a discrete set of values are called classification trees in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.
- Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.
- In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

```
model3=DecisionTreeClassifier()

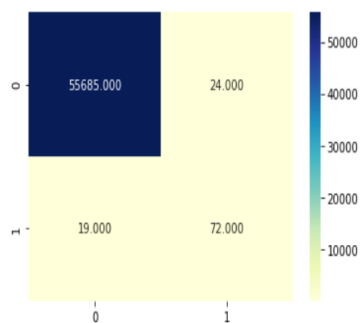
model3.fit(X_train, y_train)

y_pred = model3.predict(X_test)
acc3=metrics.accuracy_score(y_test,y_pred)
print(acc3)

0.9992293906810036
```

```
In [32]: cnf_matrix = confusion_matrix(y_test,y_pred)
```

```
In [33]: labels = [0,1]
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
plt.show()
```



Naive Bayes Algorithm:

- Naive Bayes algorithm is a probabilistic machine learning algorithm based on Bayes theorem, used in a wide variety of classification tasks.
- The simplest solutions are usually the most powerful ones, and Naïve Bayes is a good example of that. Despite the advances in Machine Learning in the last years, it has proven to not only be simple but also fast, accurate, and reliable. It has been successfully used for many purposes, but it works particularly well with natural language processing (NLP) problems.
- Bayes Theorem: It is a simple mathematical formula used for calculating conditional probabilities. Conditional probabilities is a measure of the probability of an event occurring given that another event has (by assumption, presumption, assertion, or evidence) occurred.

The formula is:

$$P(A|B) = (P(B|A) \cdot P(A)) / (P(B))$$

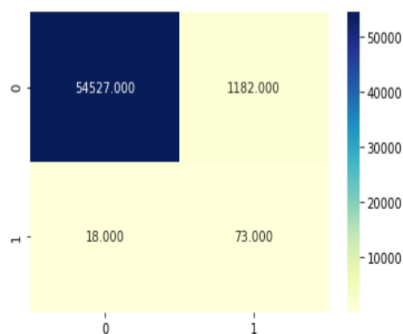
```
In [26]: from sklearn import metrics
```

```
acc2=metrics.accuracy_score(y_test,y_pred)  
print(acc2)
```

```
0.978494623655914
```

```
In [27]: cnf_matrix = confusion_matrix(y_test,y_pred)
```

```
In [28]: labels = [0,1]  
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)  
plt.show()
```

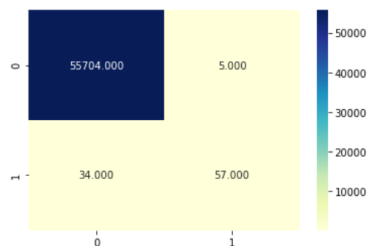


Support Vector Machine (SVM):

- SVM is a supervised machine learning model that uses classification algorithms for two-group classification problems.
- They are motivated by the principle of optimal separation, the idea that a good classifier finds the largest gap possible between data points of different classes.
- In order to perform linear classification, SVMs can efficiently perform a nonlinear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.
- More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outlier detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

```
In [37]: cnf_matrix = confusion_matrix(y_test, y_pred)
```

```
In [38]: labels = [0,1]
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
plt.show()
```



```
In [39]: acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
In [40]: model_results = pd.DataFrame([['SVM', acc, 1-rec, rec, prec, f1]],
columns = ['Model', 'Accuracy', 'FalseNegRate', 'Recall', 'Precision', 'F1 Score'])
results_testset = results_testset.append(model_results, ignore_index = True)
results_testset
```

Accuracy :

False Negative :

Recall :

Precision :

F1 Score :

Out[40]:

	Model	Accuracy	FalseNegRate	Recall	Precision	F1 Score
0	RandomForest	0.999480	0.230769	0.769231	0.897436	0.828402
1	Naive Bayes	0.978495	0.197802	0.802198	0.058167	0.108470
2	DecisionTree	0.999229	0.208791	0.791209	0.750000	0.770053
3	SVM	0.999301	0.373626	0.626374	0.919355	0.745098

1.3 Organization of Project

The technique which is developed takes transactions as input and compares it with the past data and classifies it as fraud or a safe transaction. We have three modules in our project.

- Preprocess the data
- Train the model
- Find the fraudulent transactions

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirement Gatherings

2.1.1 Software Requirements

- Programming Language : Python 3.6
- Graphical User Interface: Tkinter
- Dataset : Credit Card
- Packages : Numpy, Pandas, Matplotlib, Scikit-learn
- Tool : Jupyter Notebook

2.1.2 Hardware Requirements

- Operating System: Windows 10 / Ubuntu
- Processor : Intel Core i5

- Memory : 8 GB (RAM) and more
- Storage : 1TB

2.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python is often described as a “batteries included” language due to its comprehensive standard library.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. Many other paradigms are supported via extensions, including design by contract and logic programming.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – The code can be runned directly on the python shell which can be accessed from the terminal of the operating system.

Python also acknowledges that speed of development is important. Readable and brief code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. Python is portable which means that it can run on a wide variety of hardware platforms and has the same instance on all platforms.

Numpy

Numpy is a general-purpose array-processing library. It also has functions for working in the domain of linear algebra, fourier transforms, and matrices. Numpy stands for

Numerical python. In python we have lists that serve the purpose of arrays, but they are too slow to process. So, Numpy aims to provide an array object that is up to 50x faster than traditional Python lists.

It is the fundamental package for scientific computation with Python. It contains various features including:

- A powerful N-dimensional array object.
- Sophisticated (broadcasting) functions.
- Tools for integrating C/C++ and Fortran code.

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is a python package that provides fast, powerful, flexible and expressive data structures designed to make working with structured and time series data both easy and intuitive, built on top of the Python programming language. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn is a free software machine learning library which provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. Rather than focusing on loading, manipulating and summarising data, Scikit-learn library focuses on modeling the data. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy are conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, Mac OS and Linux. In order to run, many data scientists often use multiple versions of many packages and use multiple environments to separate these different versions. The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

A notebook document consists of rich text elements with HTML formatted text, figures, mathematical equations etc. The notebook is also an executable document consisting of code blocks in Python or other supporting languages. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Tkinter

TKinter is a python binding to the Tk GUI toolkit. It is the standard python interface to the Tk GUI toolkit and is python's de facto standard GUI. As with most other modern Tk bindings, Tkinter is implemented as a python wrapper around a complete Tcl interpreter embedded in the python interpreter. Tkinter calls are transformed into Tcl commands, which are fed to this embedded interpreter. thus making it possible to mix python and Tcl in a single application.

3. DESIGN

3.1 Introduction

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

Software design yields three levels of results:

- **Architectural Design:** It is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of the proposed solution domain.

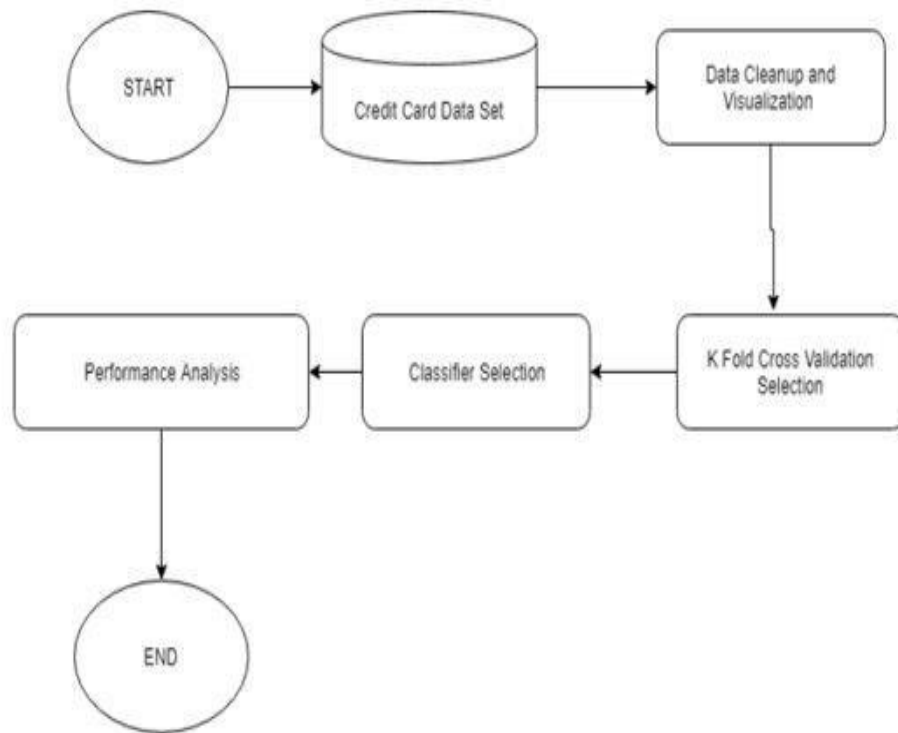
- **High-Level Design:** It breaks the ‘single entity- multiple component ‘ concept of architectural design into a less-abstracted view of subsystems and modules and depicts their interaction with each other.
- **Detailed Design:** It deals with the implementation part of what is seen as a system and its subsystems in the previous two designs. It is more detailed towards modules and their implementations.

3.2 Architecture Diagram

The main purpose of architectural diagrams is to facilitate collaboration, to increase communication, and to provide vision and guidance.

Two ways architectural diagrams can help:

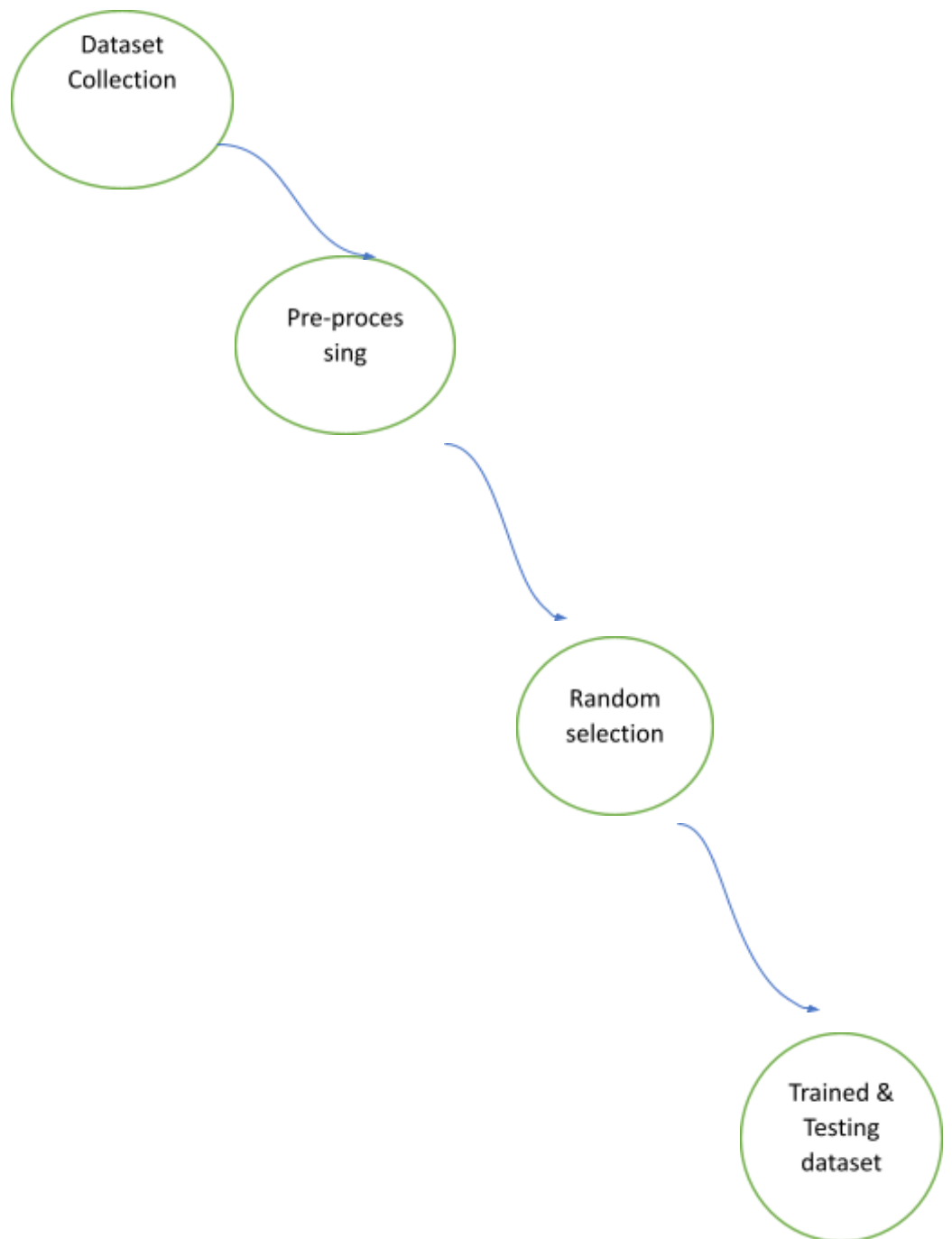
- **They help with comprehension:** Architectural diagrams show systems. displaying information visually allows the viewer to see everything at a glance, including how things interact. This is especially useful when making changes.
- **They improve communication and collaboration:** One of the main issues software engineers face is consistency. When you're working on anything that involves multiple people, there's always a risk of miscommunication and discrepancies between project teams and developers. It is crucial to standardize information which is where an architectural diagram comes in handy.



3.2.1 Data Flow Diagram

Data flow diagrams, also known as DFD, are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

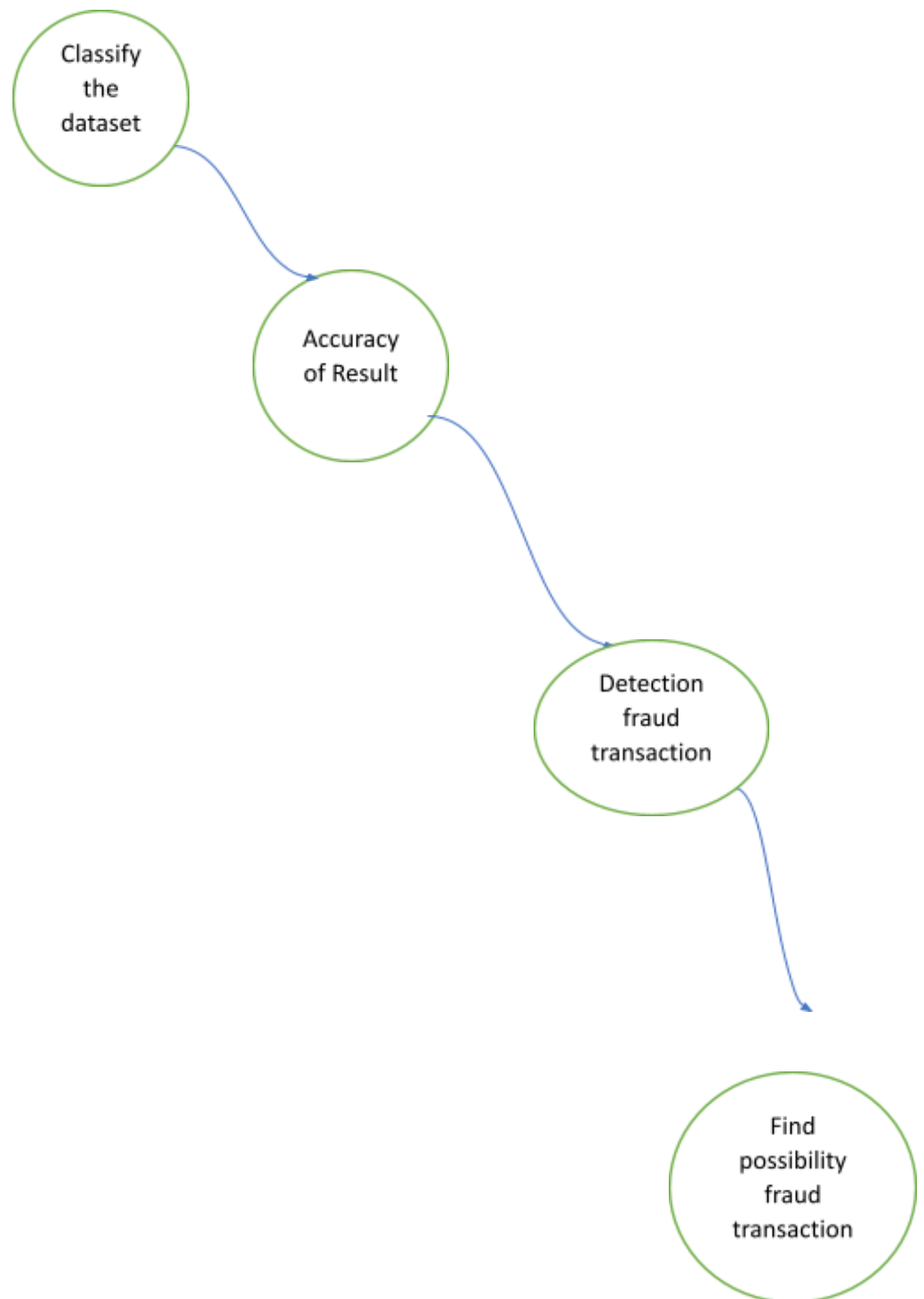
LEVEL 0



LEVEL 1



LEVEL 2



3.3 UML Diagrams

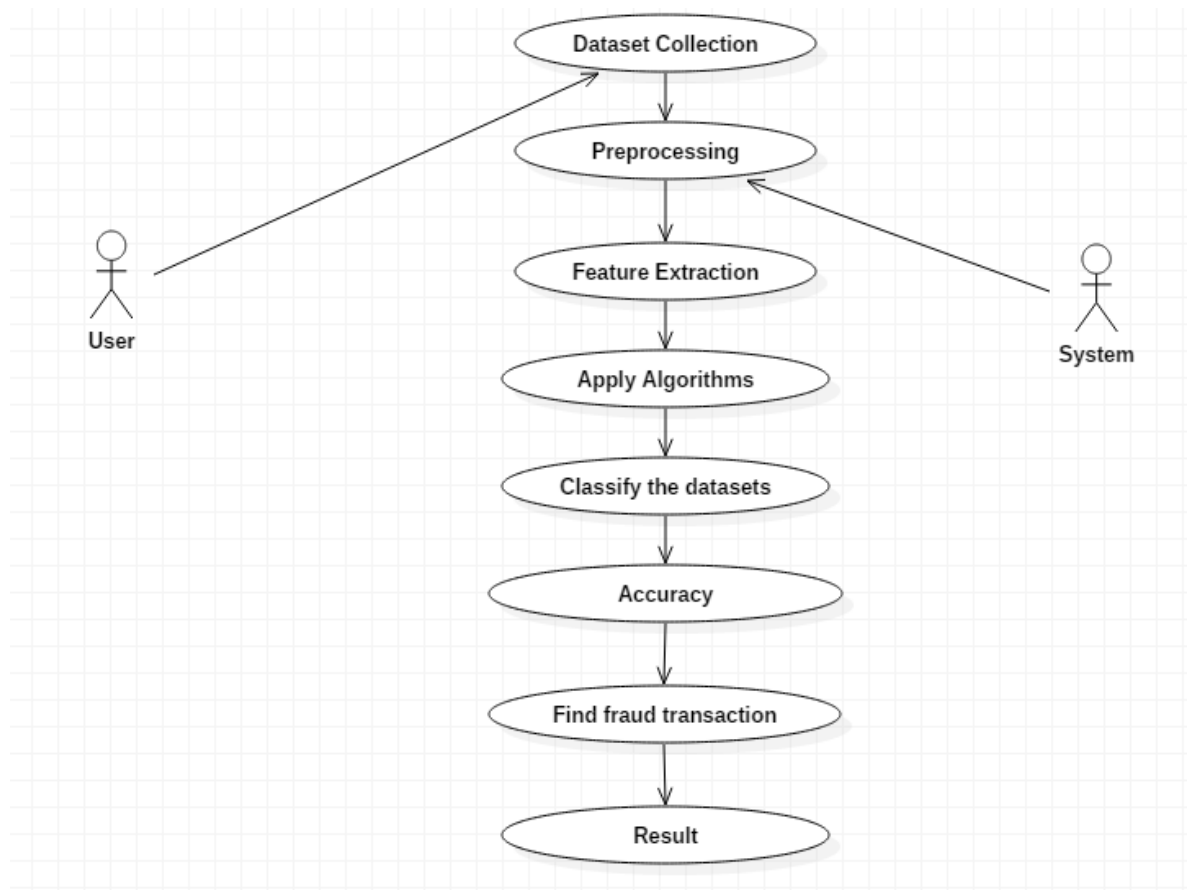
3.3.1 Use Case Diagram

A Use case diagram is a graphical representation of a user's possible interactions with a system. It shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation.

A key concept of use case modeling is that it helps design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour.

A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes some of the relationships between use cases, actors and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.



3.3.5 Class Diagram

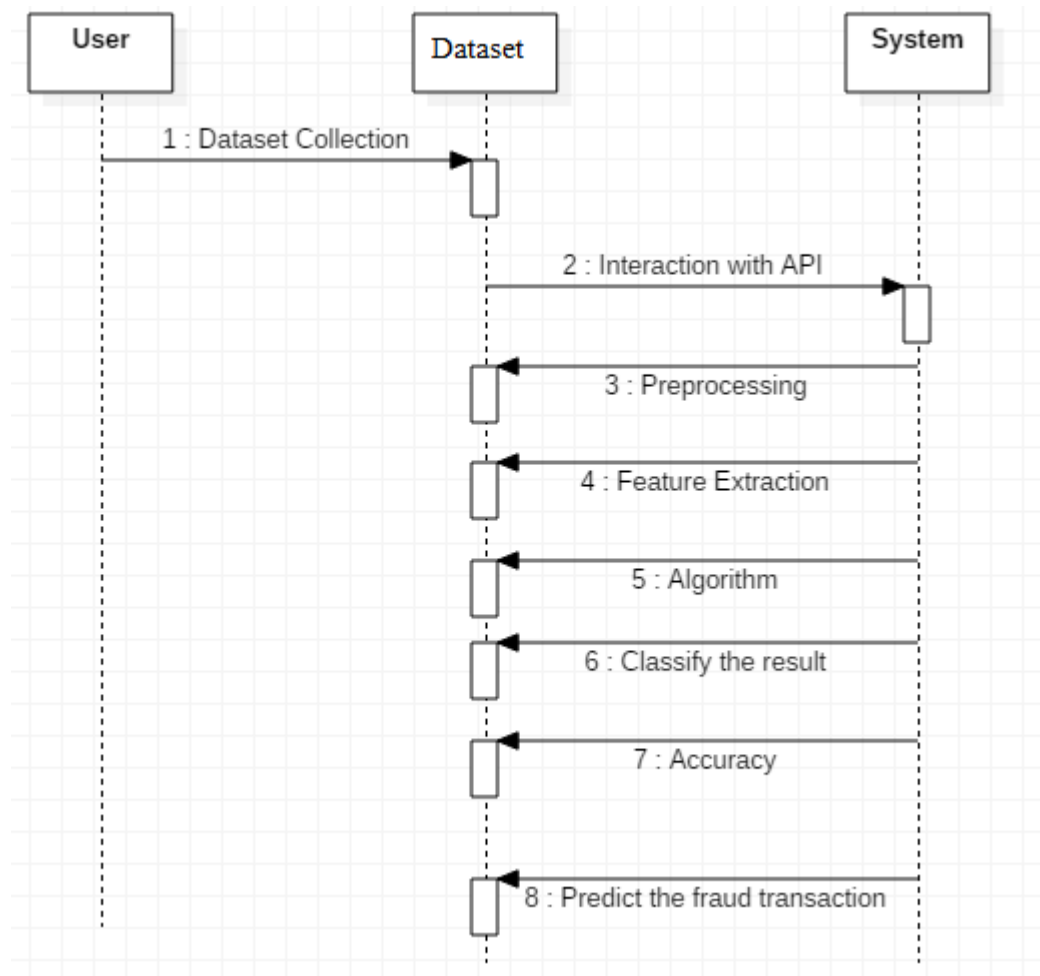
The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's:

- Classes,
- their attributes,
- operations (or methods),
- and the relationships among objects.



3.3.2 Sequence Diagram

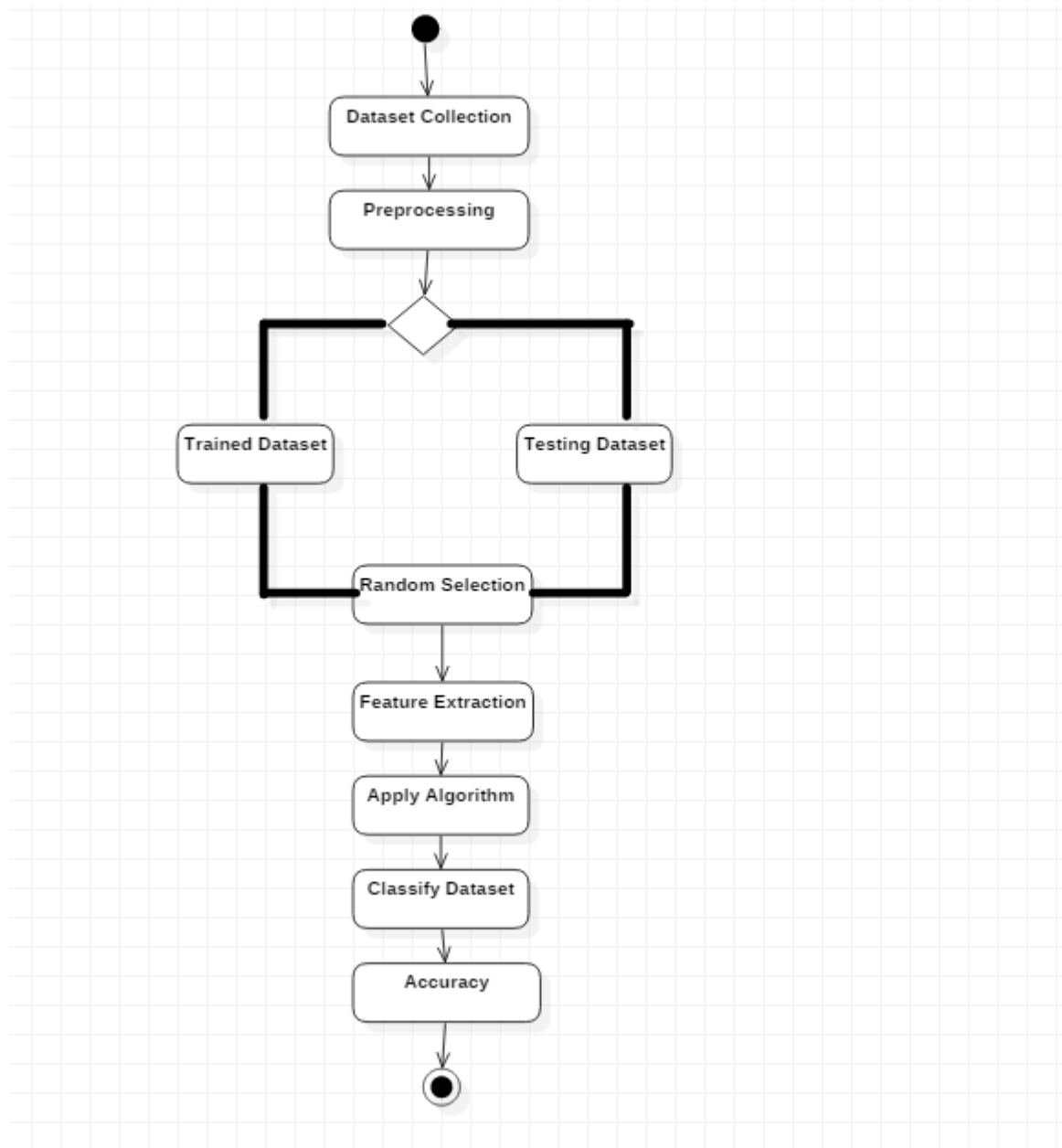
A sequence diagram is a type of interaction diagram because it describes how-and in what order-a group of objects works together. They are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.



3.3.3 Activity Diagram

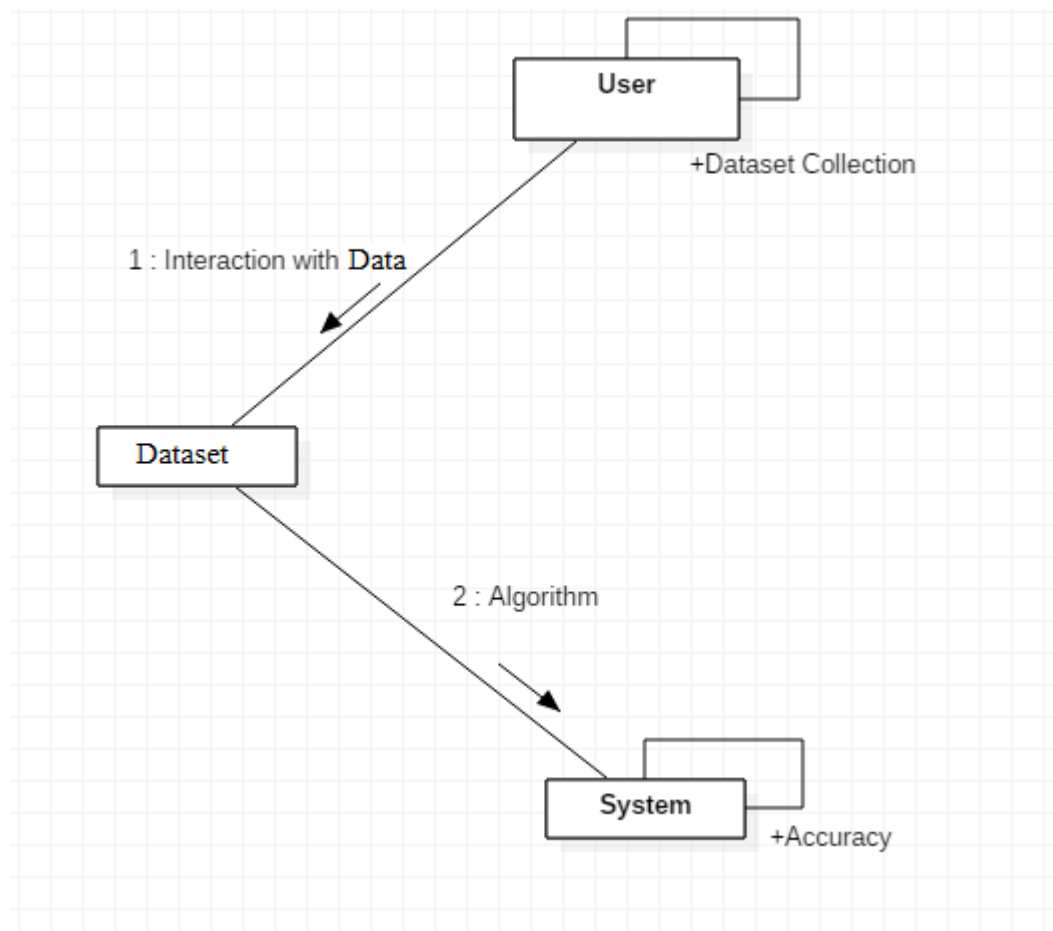
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. It is also called object oriented flowchart.

It helps in envisioning the workflow from one activity to another. It puts emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent , and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.



3.3.4 Collaboration Diagram

A collaboration diagram also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the unified modelling language. These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object. It depicts the relationships and interactions among software objects.



4.IMPLEMENTATION

4.1 Coding

Importing modules

```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
np.random.seed(2)

# Reading csv file

data = pd.read_csv('creditcard.csv')

# Data Exploration

data.info()
data.isnull()
data.head(20)

# Histogram

fig = plt.figure(figsize=(15, 20))
plt.suptitle('Histograms of Numerical Columns', fontsize=20)
for i in range(data.shape[1]):
    plt.subplot(8, 4, i + 1)
    f = plt.gca()
    f.set_title(data.columns.values[i])

    vals = np.size(data.iloc[:, i].unique())
    if vals >= 100:
        vals = 100                # limit our bins to 100 maximum

    plt.hist(data.iloc[:, i], bins=vals, color='#3F5D7D')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

```
# Correlation
```

```
data2 = data.drop(columns = ['Class']) # drop non numerical columns
data2.corrwith(data.Class).plot.bar(
    figsize = (20, 10), title = "Correlation with Class Fraudulent or Not", fontsize = 15,
    rot = 45, grid = True)
plt.show()
```

```
# Data Preprocessing
```

```
from sklearn.preprocessing import StandardScaler
data['normalizedAmount'] =
StandardScaler().fit_transform(data['Amount'].values.reshape(-1,1))
data = data.drop(['Amount'],axis=1)
```

```
data.head()
```

```
data = data.drop(['Time'],axis=1)
data.head()
```

```
X = data.iloc[:, data.columns != 'Class']
y = data.iloc[:, data.columns == 'Class']
```

```
y.head()
```

```
# Splitting the data
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state=0)
```

```
X_train.shape
```

```
X_test.shape
```

```
# Training the models
```

```
# Random Forest
```

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train,y_train.values.ravel())
y_pred = random_forest.predict(X_test)
acc1=random_forest.score(X_test,y_test)
acc1
```

```
# Confusion matrix of Random Forest
```

```
cnf_matrix = confusion_matrix(y_test,y_pred)
labels = [0,1]
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels,
yticklabels=labels)
plt.show()
```

```
# Classifier Report
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
plot_roc_curve
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
results_testset = pd.DataFrame(['RandomForest', acc, 1-rec, rec, prec, f1]),
                                columns = ['Model', 'Accuracy', 'FalseNegRate', 'Recall', 'Precision', 'F1 Score'])
results_testset
```

```
# Naive Bayes
```

```
from sklearn.naive_bayes import GaussianNB
model1 = GaussianNB()
model1.fit(X_train, y_train)
y_pred = model1.predict(X_test)
y_pred
```

```
from sklearn import metrics
acc2=metrics.accuracy_score(y_test,y_pred)
print(acc2)
```

```
# Confusion matrix of Random Forest
```

```
cnf_matrix = confusion_matrix(y_test,y_pred)
labels = [0,1]
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels,
yticklabels=labels)
plt.show()
```

```
# Classifier Report
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
plot_roc_curve
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
model_results = pd.DataFrame([[ 'Naive Bayes', acc, 1-rec, rec, prec, f1]],
                             columns = ['Model', 'Accuracy', 'FalseNegRate', 'Recall', 'Precision', 'F1 Score'])
results_testset = results_testset.append(model_results, ignore_index = True)
results_testset
```

```
# Decision Tree
```

```
from sklearn.tree import DecisionTreeClassifier  
model3=DecisionTreeClassifier()
```

```
model3.fit(X_train, y_train)  
y_pred = model3.predict(X_test)
```

```
acc3=metrics.accuracy_score(y_test,y_pred)  
print(acc3)
```

```
# Confusion matrix of Random Forest
```

```
cnf_matrix = confusion_matrix(y_test,y_pred)  
labels = [0,1]  
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels,  
yticklabels=labels)  
plt.show()
```

```
# Classifier Report
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,  
plot_roc_curve  
acc = accuracy_score(y_test, y_pred)  
prec = precision_score(y_test, y_pred)  
rec = recall_score(y_test, y_pred)  
f1 = f1_score(y_test, y_pred)
```

```
model_results = pd.DataFrame([[ 'DecisionTree', acc, 1-rec, rec, prec, f1 ]],
```

```
columns = ['Model', 'Accuracy', 'FalseNegRate', 'Recall', 'Precision', 'F1 Score'])
results_testset = results_testset.append(model_results, ignore_index = True)
results_testset
```

```
# Support Vector Machine
```

```
from sklearn import svm
```

```
model4 = svm.SVC()
```

```
model4.fit(X_train, y_train)
```

```
y_pred = model4.predict(X_test)
```

```
acc4=metrics.accuracy_score(y_test,y_pred)
```

```
print(acc4)
```

```
# Confusion matrix of Random Forest
```

```
cnf_matrix = confusion_matrix(y_test,y_pred)
```

```
labels = [0,1]
```

```
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels,  
yticklabels=labels)
```

```
plt.show()
```

```
# Classifier Report
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,  
plot_roc_curve
```

```
acc = accuracy_score(y_test, y_pred)
```

```
prec = precision_score(y_test, y_pred)
```

```
rec = recall_score(y_test, y_pred)
```

```
f1 = f1_score(y_test, y_pred)
```



```

model_results = pd.DataFrame([[ 'SVM', acc, 1-rec, rec, prec, f1]],
                             columns = ['Model', 'Accuracy', 'FalseNegRate', 'Recall', 'Precision', 'F1 Score'])
results_testset = results_testset.append(model_results, ignore_index = True)
results_testset

```

#Visualisation

Performance of algorithms based on accuracies

```

import matplotlib.pyplot as plt; plt.rcParams()
import numpy as np
import matplotlib.pyplot as plt

objects = ('RandomForest','Naive_Bayes','DecisionTree','SVM')
y_pos = np.arange(len(objects))
performance = [acc1,acc2,acc3,acc4]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Level')
plt.title('Accuracy of Algorithms')

plt.show()

print('Data set:')
print(data.columns)
for col_name in data.columns:
    #if data[col_name].dtypes == 'object' :
    unique_cat = len(data[col_name].unique())
    print(unique_cat)

```

```
print("Feature '{col_name}' has {unique_cat} categories".format(col_name=col_name,  
unique_cat=unique_cat))
```

```
print()
```

```
from tkinter import *  
from tkinter import messagebox
```

```
window = Tk()
```

```
window.title("Credit Card Fraud detection")
```

```
window.geometry('500x200')
```

```
lbl = Label(window, text="Enter account number :", width = 20)  
lbl.grid(column=0, row=0, padx=(0, 50), pady = 10)
```

```
txt = Entry(window,width=20)  
txt.grid(column=1, row=0, pady=10)
```

```
result = Label(window, text="")  
result.grid(column=1, row=2, pady=10)
```

```
def check() :  
    acc_n = txt.get()  
    if not acc_n :  
        result.configure(text="Please enter acc_no")  
    else :  
        detect(float(acc_n))
```

```
def detect(acc_n):  
    print(acc_n)
```

```

s = "Credit card acc no" + str(acc_n) + " is"

print(data["V2"][1853])
test_df = data.loc[data["V2"] == acc_n]
del test_df['Class']
#print(test_df)
test_df.reset_index(inplace = True, drop = True)
predicted = int(model1.predict(test_df)[0])
if(predicted == 0) :
    messagebox.showinfo("Safe", "Transaction successful!")
else :
    messagebox.showwarning("Alert!!", "Fraud Transaction!")

#s += "Safe" if predicted == 0 else "fraud"
#result.configure(text=s)

btn = Button(window, text="Check", command = check)
btn.grid(column=1, row=1,pady=10)

window.mainloop()

```

4.2 Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are

not limited to, the process of executing a program or application with the intent of finding software bugs.

Software Testing can also be stated as the process of validating and verifying that a software program/application/product:

- Meets the business and technical requirements that guided its design and Development.
- Works as expected and can be implemented with the same characteristics.

4.2.1 Methods

Functional Testing

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

Functional testing is centered on the following items:

- Functions: Identified functions must be exercised.
- Output: Identified classes of software outputs must be exercised.
- Systems/Procedures: system should work properly

Integration Testing

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

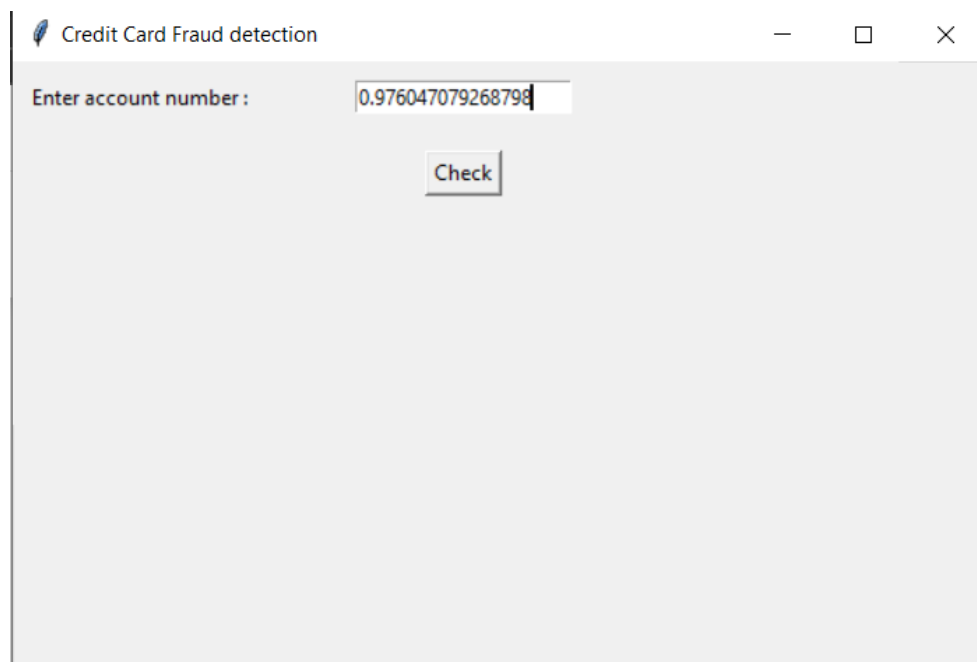
Test Case for Excel Sheet Verification:

Here in machine learning we are dealing with dataset which is in excel sheet format so if any test case we need means we need to check excel file. Later on classification will work on the respective columns of dataset .

Test Case 1 :

SL #	TEST CASE NAME	DESCRIPTION	STEP NO	ACTION TO BE TAKEN (DESIGN STEPS)	EXPECTED (DESIGN STEP)	Test Execution Result (PASS/FAIL)
1	Excel Sheet verification	Objective: There should be an excel sheet. Any number of rows can be added to the sheet.	Step 1	Excel sheet should be available	Excel sheet is available	Pass
			Step 2	Excel sheet is created based on the template	The excel sheet should always be based on the template	Pass
			Step 3	Changed the name of excel sheet	Should not make any modification on the name of excel sheet	Fail
			Step 4	Added 10000 or above records	Can add any number of records	Pass

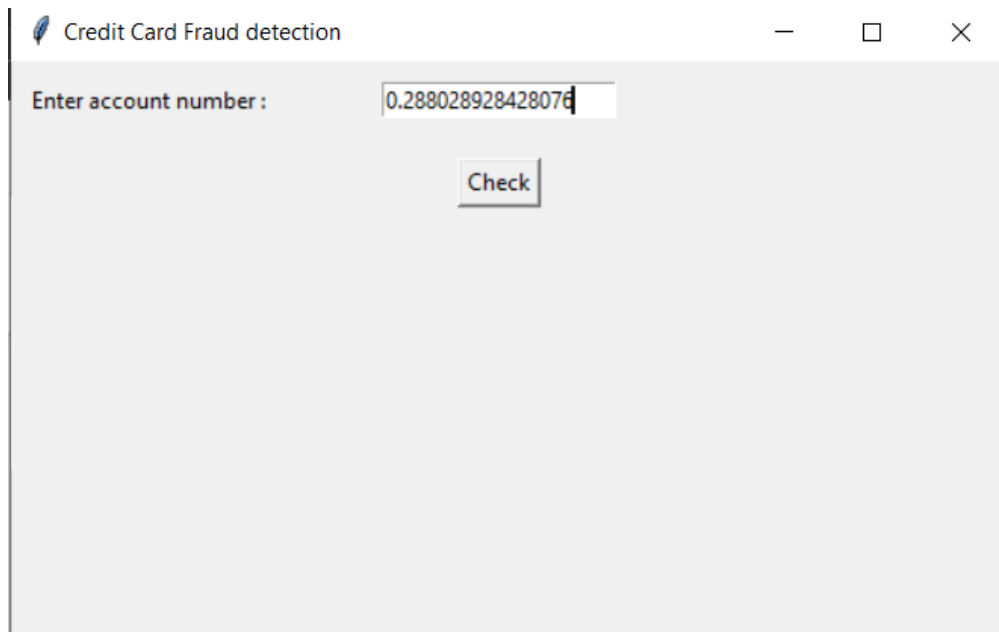
4.5 Input Screenshots



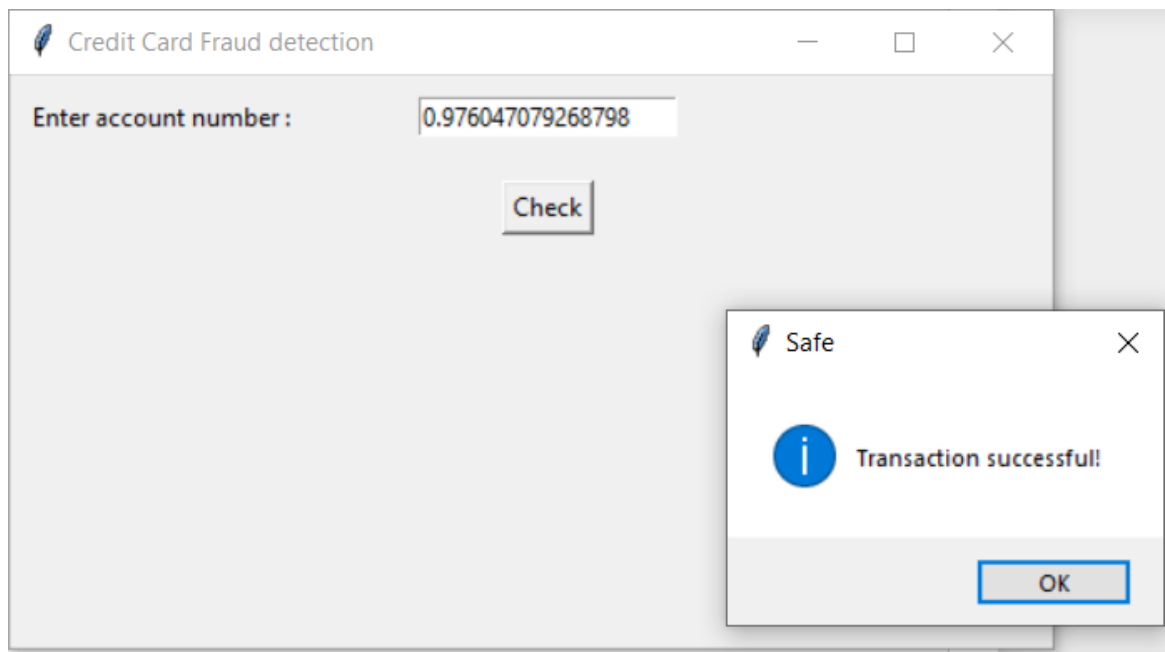
Credit Card Fraud detection

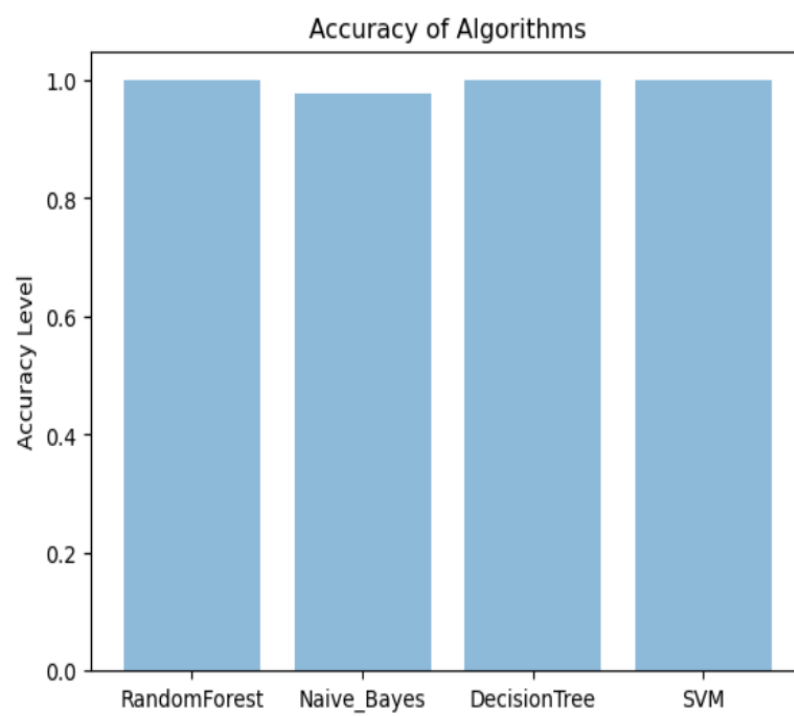
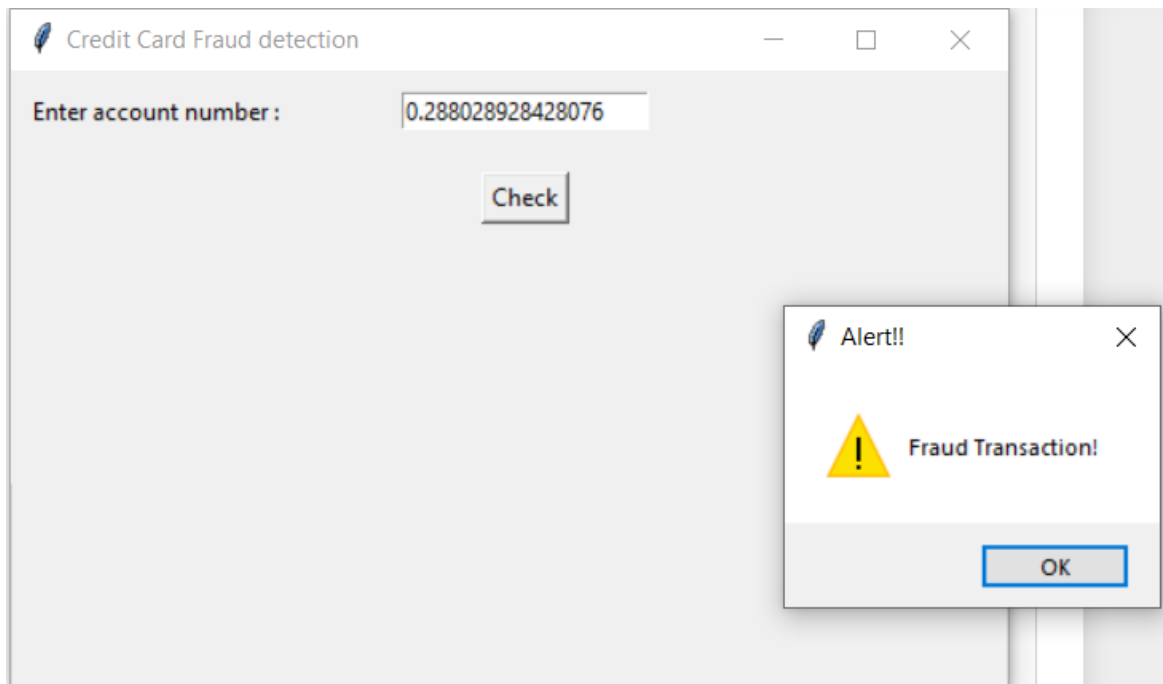
Enter account number : 0.976047079268798

Check



4.6 Output Screenshots





5.CONCLUSION AND FUTURE SCOPE

The proposed project evaluates that the Random Forest algorithm will perform better with a larger number of training data compared to the state of the other classifier , but speed during testing and application will suffer. Application of more pre-processing techniques would also help. The SVM algorithm still suffers from the imbalanced dataset problem and requires more pre-processing to give better results at the results shown by SVM is great but it could have been better if more pre-processing have been done on the data.so, in proposed work we balanced the imbalanced data with up-sampling technique during pre-processing . We review the existing works on credit card fraud prediction in three different perspectives: datasets, methods, and metrics. Firstly, we present the details about the availability of public datasets and what kinds of details are available in each dataset for predicting credit card fraud. Secondly, we compare and contrast the various predictive modeling methods that have been used in the literature for predicting, and then quantitatively compare their performances in terms of accuracy.

6. REFERENCES

- [1] P. Richhariya and P. K. Singh, "Evaluating and emerging payment card fraud challenges and resolution," *International Journal of Computer Applications*, vol. 107, no. 14, pp. 5 – 10, 2014.
- [2] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.
- [3] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," *Expert systems with applications*, vol. 41, no. 10, pp. 4915– 4928, 2014.
- [4] C. Phua, D. Alahakoon, and V. Lee, "Minority report in fraud detection: classification of skewed data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 50–59, 2004.
- [5] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.
- [6] S. Ertekin, J. Huang, and C. L. Giles, "Active learning for class imbalance problem," *The 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 823–824, 2007.
- [7] M. Wasikowski and X.-w. Chen, "Combating The Small Sample Class Imbalance problem using feature selection," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1388–1400, 2010.
- [8] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.
- [9] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical science*, pp. 235–249, 2002.

- [10] D. J. Weston, D. J. Hand, N. M. Adams, and C. Whitrow, "Plastic card fraud detection using peer group analysis," vol. 2, pp. 45–62, 2008.
- [11] E. Duman and M. H. Ozcelik, "Detecting credit card fraud by genetic algorithm and scatter search," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13057–13063, 2011.
- [12] K. Ramakalyani and D. Umadevi, "Fraud Detection of Credit Card Payment System by Genetic Algorithm," *International Journal of Scientific & Engineering Research*, vol. 3, no. 7, pp. 1–6, 2012.
- [13] P. J. Bentley, J. Kim, G.-h. Jung, and J.-u. Choi, "Fuzzy Darwinian Detection of Credit Card Fraud," pp. 1–4, 2007.
- [14] A. Srivastava, A. Kundu, S. Sural, and S. Member, "Credit Card Fraud Detection Using Hidden Markov Model," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 1, pp. 37–48, 2008.
- [15] S. Esakkiraj and S. Chidambaram, "Predictive Approach for Fraud Detection Using Hidden Markov Model," *International Journal of Engineering Research & Technology*, vol. 2, no. 1, pp. 1–7, 2013.
- [16] J. S. Mishra, S. Panda, and A. K. Mishra, "A Novel Approach for Credit Card Fraud Detection Targeting the Indian Market," *International Journal of Computer Science*, vol. 10, no. 3, pp. 172–179, 2013.
- [17] A. Brabazon, J. Cahill, P. Keenan, and D. Walsh, "Identifying Online Credit Card Fraud using Artificial Immune Systems," *IEEE Congress on Evolutionary Computation*, pp. 1 – 7, 2010.
- [18] N. Wong, P. Ray, G. Stephens, and L. Lewis, "Artificial immune systems for the detection of credit card fraud: an architecture, prototype and preliminary results," *Information systems*, vol. 22, pp. 53–76, 2012.
- [19] D. Sanchez, M. A. Vila, L. Cerda, and J. M. Serrano, "Association rules applied to credit card fraud detection," *ScienceDirect*, vol. 36, pp. 3630– 3640, 2009.
- [20] Y. Sahin, S. Bulkan, and E. Duman, " A cost-sensitive decision tree approach for fraud detection," *Expert Systems with Applications*, vol. 40, pp. 5916–5918, 2013.
- [21] A. C. Bahnsen, A. Stojanovic, and D. Aouada, "Cost Sensitive Credit Card Fraud Detection using Bayes Minimum Risk," *12th International Conference on Machine Learning and Applications*, pp. 333–338, 2013.
- [22] A. E. Pasarica, "Card fraud detection using learning machines," *The Bulletin of the Polytechnic Institute of Jassy*, pp. 29 – 45, 2014.

- [23] Y. Sahin and E. Duman, "Detecting Credit Card Fraud by Decision Trees and Support Vector Machines," International Multiconference of Engineers and computer scientists, vol. 1, pp. 442–447, 2011.
- [24] V.R.Ganji and S.N.P.Mannem, "Credit Card Fraud Detection Using Anti-k nearest neighbor algorithm," International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 06, pp. 1035–1039, 2012.
- [25] S. Ghosh and D. L. Reilly, "Credit Card Fraud Detection with a Neural Network," Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences, 1994, pp. 621–630, 1994.
- [26] R.Dorronsoro, F.Ginel, S.Carmen, and C.S.Cruz, "Neural Fraud Detection in Credit Card Operations," IEEE Transactions on Neural Networks, vol. 8, no. 4, pp. 827–834, 1997.
- [27] V. Zaslavsky and A. Strizhak, "Credit card fraud detection using selforganizing maps," Information and Security, vol. 18, pp. 48–63, 2006.
- [28] F. N. Ogwueleka, "Data Mining Application in Credit Card Fraud Detection System," Journal of Engineering Science and Technology, vol. 6, no. 3, pp. 311–322, 2011.
- [29] R. Patidar and L. Sharma, "Credit Card Fraud Detection Using Neural Network," International Journal of Soft Computing and Engineering, no. May, pp. 13–14, 2011.
- [30] M. Syeda, Y.-q. Zhang, Y. Pan, and C. Science, "Parallel Granular Neural Networks for Fast Credit Card Fraud Detection," Proceedings of the IEEE International Conference on Fuzzy Systems., vol. 1, pp. 572–577, 2002.
- [31] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, "Credit card fraud detection using bayesian and neural networks," Interactive image guided neurosurgery. American Association Neurological Surgeons, pp. 261–270, 1993.