FRAUDULENT TRANSACTION DETECTION SYSTEM

Devi Priya Bijosh Mohan

c1056531

APPENDIX

INTRODUCTION

Cashless payments have gained importance in recent decades because of the rapid development of the widespread Internet. Due to the easiness, most people opt for cashless payments for retail purchases, restaurants, transportation, insurance, internet auction etc. It increases the chances of being engaged with a fraudster online. So, the ability to identify fraudulent transactions became a vital part of this modern technological era. Because of this, the need for software has been massively increased throughout time. The use of credit cards is one of the major types of cashless payment which has obtained large popularity. This project is a fraudulent transaction detection system that aims to implement functionalities such as fraudulence check, get details of transactions, check fraudulence using tagged keywords, future usable statistical functions etc.

PROBLEM ANALYSIS

The internet is a location where people from all over the world may communicate with one another via various platforms. Ecommerce has become extensively accessible to the public, allowing fraudsters to fool them and steal their money in a variety of methods. According to reports, the majority of scams occur when people use their credit or debit cards to make purchases. To combat these types of fraudulent transactions, a solution must be created, and clients must be aware of the processes involved in preventing them. Banking software has been developed and scaled up to a significant extent to prevent these types of scams with an increased layer of security. This project attempts to create a fraud detection system that uses multiple tagged keywords and statistical methods to detect fraudulent transactions. The dataset is initially provided as input, and the required values are retrieved and fed into the implemented functions to determine the output values.

SOLUTION OF REQUIREMENTS

The system's functional and non-functional requirements are both addressed in the requirements solution. The functional requirement outlines the functional parts of the system that are being built, while the non-functional requirement refers to how the system should interact with end-users.

Functional Requirements

1. The system must accept the provided dataset as the input.
2. The required values must be extracted according to inputs needed for each functionality.
3. The intended functionalities include, finding a fraudulent transaction and usage of statistical functions such as minimum, maximum, mean, variance, centroid, distance from centroid etc. to get a better view of data related to each transaction and future statistical analysis purpose.
4. Additional functionalities such as computing distance between any two transactions of any user or a single user.
5. The errors encountered must be handled properly including exceptions, key errors etc.
6. The system should be satisfied with modularity by the 3 main modules each having different purposes.

Non-functional requirements

1. The user interface must be user-friendly with proper options shown to the end-user.
2. The error messages must be meaningful and understandable.

IMPLEMENTATION OF SOLUTION

The system has been implemented with 3 modules namely 3 modules namely *load_dataset_module, user_statistics_module* and test module. The input dataset(s) are transaction.txt, fraud-description.txt, and description.txt and the datasets are added to dictionaries in the load_dataset_module. The values extracted from these files are used as inputs for functions such as **maximumMinimumTransaction, getCentroid, getStandardDeviation, getVariance, isFraud, listFraudTransaction, distanceTransactionFromCentroid, dataWithTransactionId, verifyTransactionOfSameUser, distanceBtnTwoTransactions, distanceBtnTwoTransactionsOfAUser, distanceTransactionFromCentroid** in the user_statistics_module.

The user interface is implemented in the test_module with different options provided. The required modules are imported as python file objects. The statistic function imports the load_dataset_module for the function such as the test_module imports user_statistics_module to perform the functions in the user interface. Each module has its individuality and communicates to another module efficiently. From the execution file, we can execute all these functions by importing the test_module.

PROGRAM EXECUTION

The entire project is divided into the following 3 main modules.

load_dataset_module: This contains the function to return the dictionary which takes data from the transaction.txt dataset and also lists that contains the fraudulent tags.

user_statistics_module : It contains all the statistics functions such as maximumMinimumTransaction, getCentroid, getStandardDeviation, getVariance, isFraud, listFraudTransaction, distanceTransactionFromCentroid, distanceBtnTwoTransactions, distanceBtnTwoTransactionsOfAUser, distanceTransactionFromCentroid.

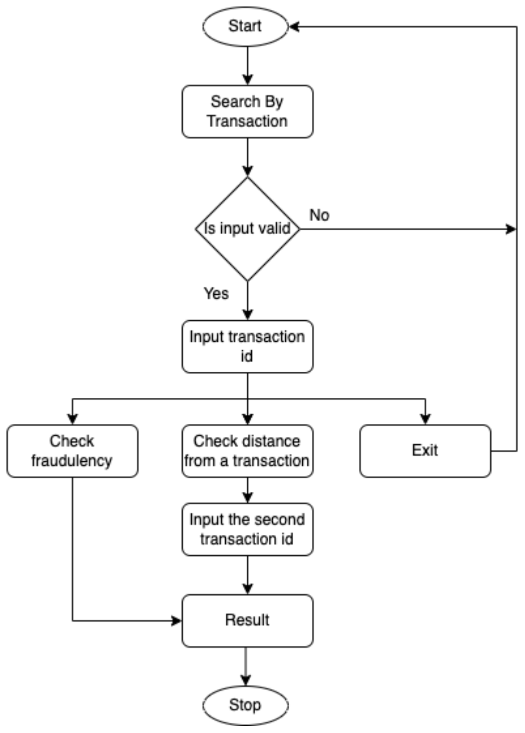
test_module: It contains the user interface functions. In the test module, we have 2 main options based on transaction id and user id, which allows a user to search by transaction/ user id to make the user interface simpler. It import statistics function and load data function by python file objects. The functions of the 2 options are;

- Checking fraudulency and distance from another transaction, with inputting a transaction id.
- Checking minimum and maximum transaction, centroid, standard deviation, variance, list of fraud transaction, the distance of the transaction from the centroid and verify 2 transactions of the same user by inputting user Id.

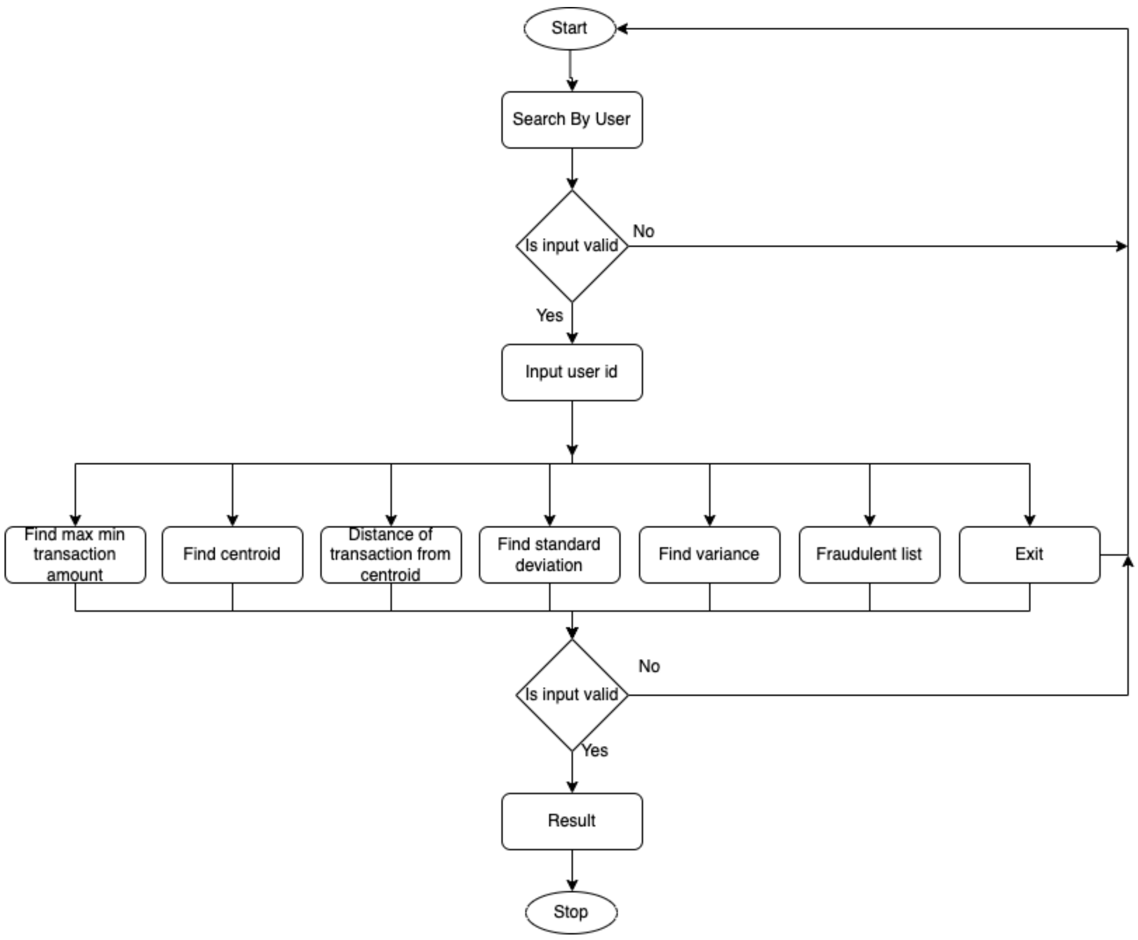
Steps to Execute the project:

1. Unzip the souce code
2. Launch ‘execution_file.ipynb in Jupiter notebook
3. Run main()
4. Enter the inputs according to the menu.

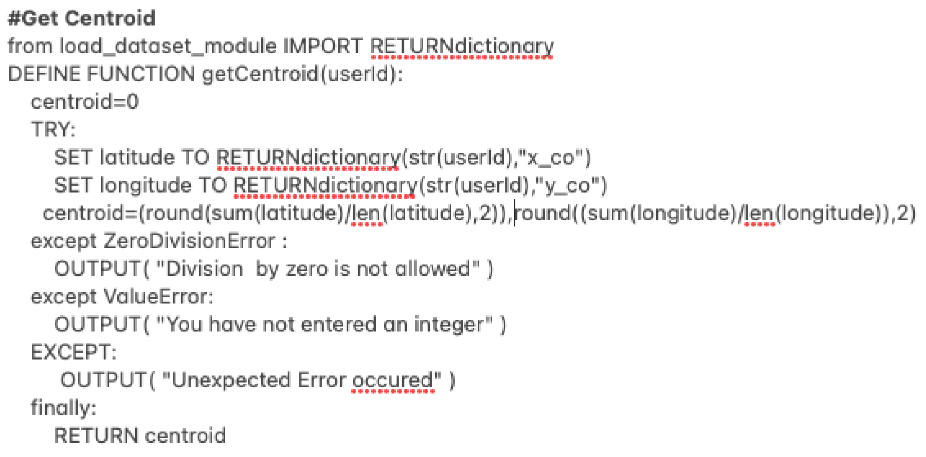
FLOWCHART : SEARCH BY TRANSACTION



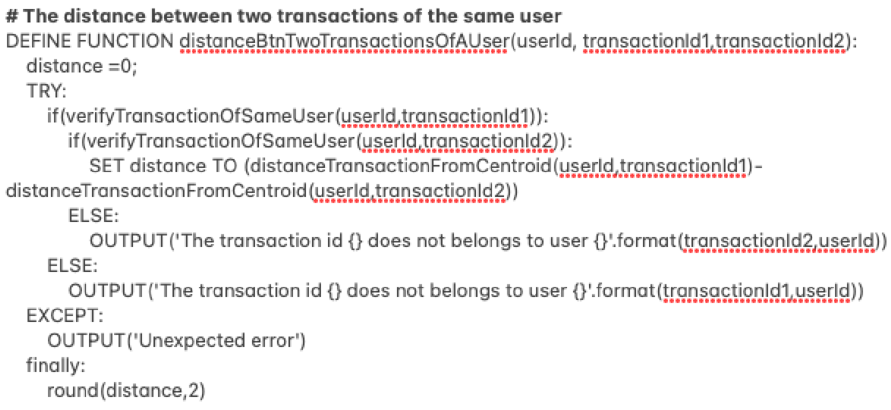
FLOWCHART : SEARCH BY TRANSACTION



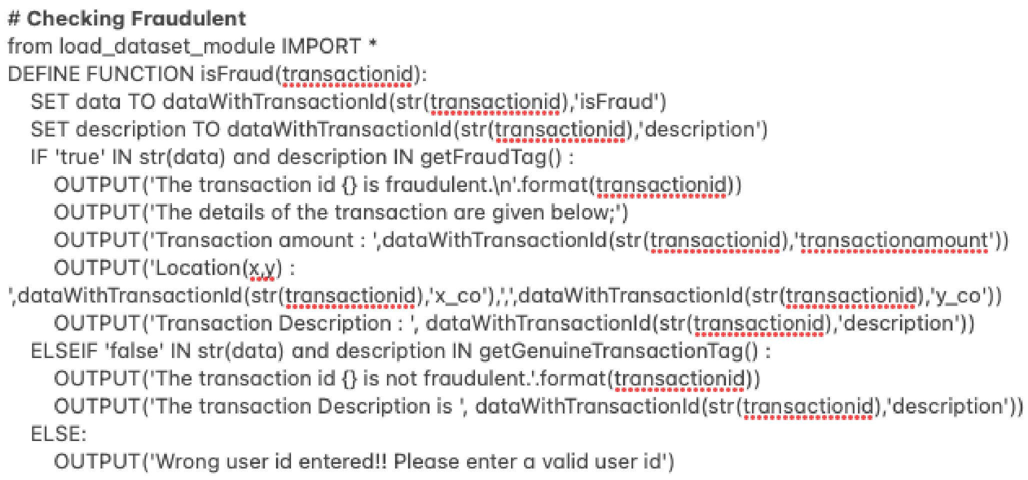
MODULE PSEUDOCODE: 3 statistical functions



```
#Get Centroid
from load_dataset_module IMPORT RETURNdictionary
DEFINE FUNCTION getCentroid(userId):
  centroid=0
  TRY:
    SET latitude TO RETURNdictionary(str(userId),"x_co")
    SET longitude TO RETURNdictionary(str(userId),"y_co")
   centroid=(round(sum(latitude)/len(latitude),2)),round((sum(longitude)/len(longitude)),2)
  except ZeroDivisionError :
    OUTPUT( "Division  by zero is not allowed" )
  except ValueError:
    OUTPUT( "You have not entered an integer" )
  EXCEPT:
     OUTPUT( "Unexpected Error occured" )
  finally:
    RETURN centroid
```



```
# The distance between two transactions of the same user
DEFINE FUNCTION distanceBtnTwoTransactionsOfAUser(userId, transactionId1,transactionId2):
  distance =0;
  TRY:
    if(verifyTransactionOfSameUser(userId,transactionId1)):
      if(verifyTransactionOfSameUser(userId,transactionId2)):
        SET distance TO (distanceTransactionFromCentroid(userId,transactionId1)-
distanceTransactionFromCentroid(userId,transactionId2))
      ELSE:
        OUTPUT('The transaction id {} does not belongs to user {}'.format(transactionId2,userId))
    ELSE:
      OUTPUT('The transaction id {} does not belongs to user {}'.format(transactionId1,userId))
  EXCEPT:
    OUTPUT('Unexpected error')
  finally:
    round(distance,2)
```



```
# Checking Fraudulent
from load_dataset_module IMPORT *
DEFINE FUNCTION isFraud(transactionid):
  SET data TO dataWithTransactionId(str(transactionid),'isFraud')
  SET description TO dataWithTransactionId(str(transactionid),'description')
  IF 'true' IN str(data) and description IN getFraudTag() :
    OUTPUT('The transaction id {} is fraudulent.\n'.format(transactionid))
    OUTPUT('The details of the transaction are given below;')
    OUTPUT('Transaction amount : ',dataWithTransactionId(str(transactionid),'transactionamount'))
    OUTPUT('Location(x,y) :
',dataWithTransactionId(str(transactionid),'x_co'),',',dataWithTransactionId(str(transactionid),'y_co'))
    OUTPUT('Transaction Description : ', dataWithTransactionId(str(transactionid),'description'))
  ELSEIF 'false' IN str(data) and description IN getGenuineTransactionTag() :
    OUTPUT('The transaction id {} is not fraudulent.'.format(transactionid))
    OUTPUT('The transaction Description is ', dataWithTransactionId(str(transactionid),'description'))
  ELSE:
    OUTPUT('Wrong user id entered!! Please enter a valid user id')
```

REFLECTION AND CONCLUSION

The system is currently being developed for a user who can perform fraud detection and statistics searches using his user id/transaction id. Multiple user levels views, such as ordinary credit card customers and authorised bank professionals, are not implemented in this software. Other factors and statistical analysis can be looked at further, but they aren't part of the current prototype. The development of this project aids in the understanding of the project's needs, resulting in modularity, usability, and resilience. Furthermore, it makes me consider and recognise the current relevance of such a system.