Anti-Money Laundering using Python and Keras

A Project Report submitted in partial fulfilment for the award of

MBA (FinTech)

Submitted by

Abhigna Sai Bathina (121923901001)

Under the guidance of

Mr. Leben Johnson

Associate Professor

****

GITAM INSTITUTE OF MANAGEMENT

GITAM UNIVERSITY

(U/s 3 of UGC Act)

VISAKHAPATNAM

(2019-21)

DECLARATION

I, the undersigned, hereby declare that, the project titled “Anti-money Laundering using Python and Keras” submitted to GITAM Institute of Management, GITAM University for the award of the Masters of Business Administration (FinTech) is the original work done by us, under the guidance of Mr.Leben Johnson and Mr.Divakar Allavarapu, GITAM Institute of Management. The empirical findings in this report are based on the work done by us. It was not copied from any other project. The project has not been submitted to any Substitute/University for the award of any Diploma/Degree.

Date: - Abhigna Sai Bathina

Place: - Visakhapatnam 121923901001

CERTIFICATE

This is to certify that the project entitled, “Anti-money Laundering using Python and Keras”, is a bonafide work done by ABHIGNA SAI BATHINA(121923901001) is submitted in partial fulfillment for the Masters of Business Administration (FinTech) of GITAM University. It has not been submitted for the award of any diploma/degree in any other Institution/University.

Date: Mr. Leben Johnson

Place: Visakhapatnam Associate Professor

INDEX

Contents Page-No

1. Introduction 5 - 6

2. Dataset 7

3. Python and Keras 8

4. Packages Used 9

5. Source Code 10-18

6. Conclusion 18

**Money Laundering**

Money laundering is an illegal process of concealing the origins of money obtained illegally by passing it through a complex sequence of banking transfers or commercial transactions. The overall scheme of this process returns the money to the launderer in an obscure and indirect way. A complete money laundering operation will often involve several schemes as the money is moved around to avoid detection.

There are three stages in [a typical money laundering process](https://www.casino.org/blog/how-money-laundering-really-works/) they are Placement which is the act of depositing money into the financial system. Then layering, which is the source of the proceeds disguised by creating complex layers of financial transactions to obscure any audit trail. The final one is Integration where laundered money is integrated into the legitimate financial system.

**Anti-money Laundering:**

Anti-money laundering (AML) is a term mainly used in the financial and legal industries to describe the legal controls that require [financial institutions](https://en.wikipedia.org/wiki/Financial_institution) and other regulated entities to prevent, detect, and report money laundering activities. Anti-money laundering software is used in the finance and [legal](https://en.wikipedia.org/wiki/Legal) industries to meet the legal requirements for [financial institutions](https://en.wikipedia.org/wiki/Financial_institution) and other regulated entities to prevent or report [money laundering](https://en.wikipedia.org/wiki/Money_laundering) activities.

The main objective of these software is to identify whether a traction is fraudulent or not. Traditional AML systems are based on rules engines, helping companies to identify suspicious activity. These rules-based systems create alerts and flag transactions as suspicious based on pre-determined rules. The main criteria for identifying that in our project is the cash getting out of the financial system after a complex set of transactions.

The issue with relying on a rule-based system to tackle money laundering is these traditional systems are rigid and cannot adapt to continuously changing data. Additionally, a great number of false positives are included in alerts which are not identified to be as such until much later in the process. Therefore, the goal is to find a way to reduce the number of false positives which can be achieved by applying machine learning techniques such as Neural Networks.

**Deep Learning and Neural Networks:**

Deep learning is an increasingly popular subset of machine learning. Deep learning models are built using neural networks. Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They help us cluster and classify. A neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training. Then the model spits out a prediction.

Keras is a user-friendly neural network library written in Python in which we used the sequential model for our prediction. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. The activation function we will be using is ReLU or Rectified Linear Activation. An activation function allows models to take into account nonlinear relationships.

**Dataset**

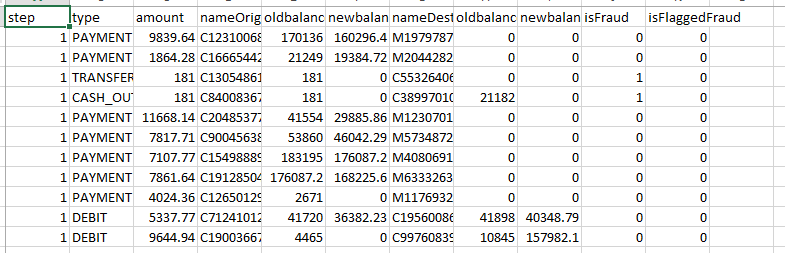
This is a Synthetic financial dataset for fraud detection from Kaggle.com.

There is a lack of publicly available datasets on financial services, as there is a concept of confidentiality involved in it. But there is a demand for publicly available datasets of financial transactions so, this synthetic dataset is generated using the simulator called PaySim as an approach to such a problem.

PaySim uses aggregated data from the private dataset to generate a synthetic dataset that resembles the normal operation of transactions and injects malicious behaviour to later evaluate the performance of fraud detection methods.

PaySim simulates mobile money transactions based on a sample of real transactions extracted from one month of financial logs from a mobile money service implemented in an African country. The original logs were provided by a multinational company, who is the provider of the mobile financial service which is currently running in more than 14 countries all around the world.

The dataset looks like this



There are 5 types of financial records categories are CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.

**Python and Keras:**

The platform being used for this model on Anti Money Laundering is Python Anaconda Platform. Python is a widely used high-level programming language for general-purpose programming. Apart from being open source programming language, python is a great object-oriented, interpreted, and interactive programming language. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems. Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. It was developed to make implementing deep learning models as fast and easy as possible for research and development.

**Keras Installation:**

Keras runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. Itis a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. Keras follows the best practices for reducing cognitive loads.Keras integrates with lower-level deep learning languages (in particular TensorFlow), it enables you to implement anything you could have built in the base language.

For the installation of the Keras Library we need to first pip install it in the anaconda prompt

>conda install –c conda-forge keras

>conda install –c conda-forge imbalanced-learn

Then in Python console

>import Keras

**Packages Used:**

* pandas: pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
* sklearn: The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
* imblearn.over\_sampling: This package is used to deal with imbalanced datasets. It provides methods to under or over sample the dataset. Class to perform over-sampling done using SMOTE. This object is an implementation of SMOTE - Synthetic Minority Over-sampling Technique.
* keras.models: There are two main types of models available in Keras, the Sequential model and the Model class used with the functional API. In addition to these two types of models, you may create your own fully-customizable models by sub-classing the Model class and implementing your own forward pass in the call method.
* keras.layers: Keras layers are the fundamental building block of keras models. Layers are created using a wide variety of layer\_ functions and are typically composed together by stacking calls to them using the pipe %>% operator. This is a flattened list of the layers comprising the model.

**Source Code:**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from imblearn.over\_sampling import SMOTE

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

from keras.models import Sequential

from keras.layers import Dense

#Importing the dataset as a dataframe

df = pd.read\_csv("amlds.csv")

df.head()

df.columns

#removing the unwanted columns

df.drop('nameOrig', axis=1, inplace=True)

df.drop('nameDest', axis=1, inplace=True)

df.drop('isFlaggedFraud', axis=1, inplace=True)

#Checking for any null values

print('Null Values =',df.isnull().values.any())

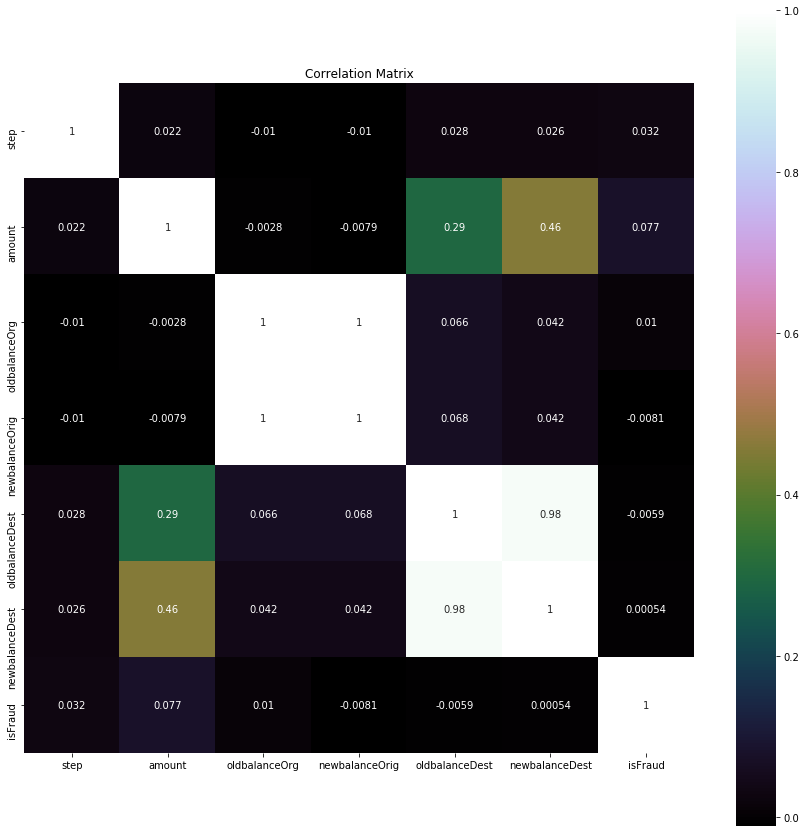
#correlation matrix to check multicolinearity between the variables

correlation = df.corr()

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

plt.title('Correlation Matrix')

sns.heatmap(correlation, vmax=1, square=True,annot=True,cmap='cubehelix')

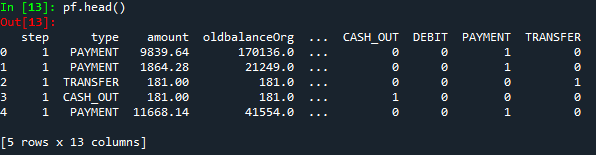


#creating dummy variables for categorical values and dropping the original

dum = pd.get\_dummies(df['type'])

pf = pd.concat([df,dum],axis=1)

pf.drop(['type'],axis=1, inplace=True)



#Understanding the ratio of fraud and not

bf = pf.sample(n=20000)

bf.isFraud.value\_counts().plot.bar()

print(bf.isFraud.value\_counts())

#Splitting the data into training and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(bf.drop(['isFraud'],axis=1), bf['isFraud'], test\_size=0.3, random\_state=0)

print("Before OverSampling, counts of label '1': {}".format(sum(y\_train==1)))

print("Before OverSampling, counts of label '0': {} \n".format(sum(y\_train==0)))

# resampling the training data

sm = SMOTE(random\_state=10)

x\_train\_res, y\_train\_res = sm.fit\_sample(X\_train, y\_train)

print('After OverSampling, the shape of train\_X: {}'.format(x\_train\_res.shape))

print('After OverSampling, the shape of train\_y: {} \n'.format(y\_train\_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y\_train\_res==1)))

print("After OverSampling, counts of label '0': {}".format(sum(y\_train\_res==0)))

# Feature scaling

sc = StandardScaler()

x\_train\_scaled = sc.fit\_transform(x\_train\_res)

x\_test\_scaled = sc.transform(X\_test)

# Initializing the model using keras

model = Sequential()

# Adding the input layer and the first hidden layer in the Neural Network model

model.add(Dense(input\_dim=11, output\_dim = 6, init = 'uniform', activation = 'relu'))

#Adding the second hidden layer

model.add(Dense(output\_dim = 6, init = 'uniform', activation = 'relu'))

# Adding the output layer

model.add(Dense(output\_dim = 1, init = 'uniform', activation = 'sigmoid'))

# Compiling and fitting the model

model.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

model\_info = model.fit(x\_train\_scaled, y\_train\_res, batch\_size = 10, nb\_epoch = 10)

history = model.fit(x\_train\_scaled, y\_train\_res, validation\_split=0.2, epochs=10, verbose=1)

# Plot training & validation accuracy values

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

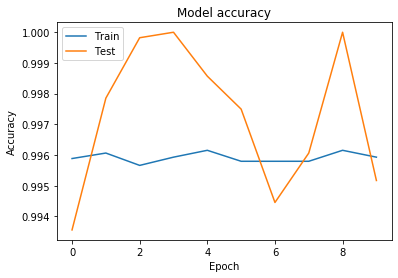
plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()



# Plot training & validation loss values

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

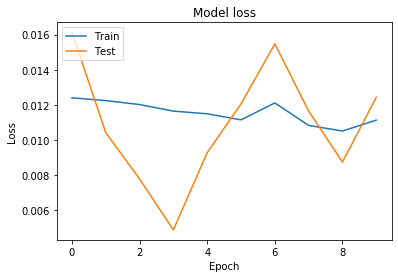
plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()



# Predicting the test results

y\_pred = model.predict\_classes(x\_test\_scaled)

acc = accuracy\_score(y\_test,y\_pred)\*100

print('Accuracy:',round(acc,2))

print("counts of label '1': {}".format(sum(y\_pred==1)))

print("counts of label '0': {} \n".format(sum(y\_pred==0)))

# Generating the Confusion matrix and Classification report

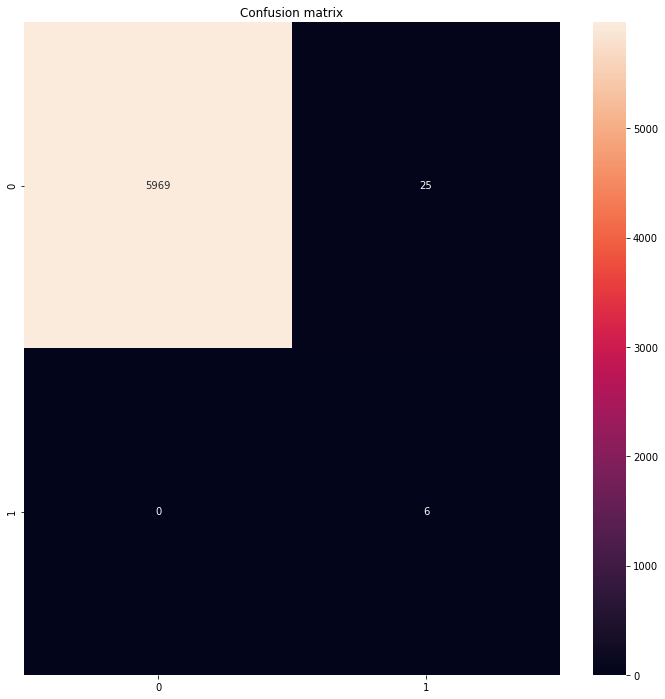
mat=confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix', '\n', mat, '\n')

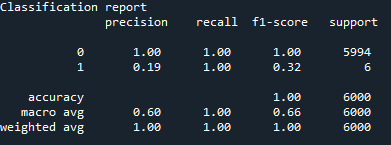
plt.figure(figsize=(12, 12))

sns.heatmap(mat, annot=True, fmt="d");

plt.title("Confusion matrix")



print('Classification report', '\n', classification\_report(y\_test, y\_pred), '\n')



Conclusion: In this way we used the sequential model in Keras in order to build a model for the prediction of values. The values predicted are of accuracy 99 percent we is almost similar to the original dataset. The predicted values has minimized number of false positives.