2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence) | 978-1-6654-3701-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/Confluence52989.2022.9734203

Counterfeit Regulation through Machine Learning Approach and Deployment in Dockers

Ajit Kumar Rout
Department of Information Technology
GMR Institue of Technology
Rajam, Andhra Pradesh, India
ajitkumar.rout@gmrit.edu.in

Abhishek Shety
Department of Information Technology
GMR Institue of Technology
Rajam, Andhra Pradesh, India
abhisek.s@gmrit.edu.in

Koushik Modekurti
Department of Information technolgy
GMR Institue of Technology
Rajam, Andhra Pradesh, India
koushikmodekurti00@gmail.com

Abstract - Money is very essential in today's world to lead a good & prosperous life. Money Mints are responsible for the printing of banknotes for a Nation. Dissimilitude here mentioned refers to unrestrained reproduction of currency with the help of advanced technology accessible to all. The act of manufacturing replicated currency without the authenticity of the State is called Counterfeiting. Fake Notes are being produced with many features that can't be easily recognized by us and are similar to original Bank Notes. Therefore, an efficient mechanism is to be engineered for regulating the Counterfeit of Money. This paper implements Wavelet Transformed images of banknotes. Supervised Learning Classifiers, Random Forest & Naïve Bayes are compared & used to predict original one. Performance Evaluation is based on parameters such as Accuracy. A User-Friendly Interface is designed using Flask, Flagger to view the predictions made by the Model and Docker for environment standardization.

Keywords - Counterfeit, Wavelet Transformation, Random Forest, Naïve Bayes, Flask, Docker

INTRODUCTION

Money is a widely accepted and standard medium for exchange while the occurrence of a transaction. A transaction can happen between any two entities. These transactions have been occurring in the world since ancient times. Money is also sometimes referred to as Currency, which is a very essential economic unit in the financial world & is an interchangeably used term. Currency is usually exchanged for goods & services in return. Currency can come in various forms either coins or notes. The origin of currency can be traced back to the primitive period where coins were the most common form of currency. Karshapanas were some of the earliest coins in the Indian subcontinent which were used back in the 6th Century B.C. Natural Forces like Sun, Moon were imbibed on them [1]. Coins being made of metal from core had high endurance & thus were extensively used. Then, a transition towards durable lightweight substance began, thereby evolved Notes. Chinese were the first in the world to flourish the usage of notes around (AD 960 - 1279). With a more convenient mode of trading commodities & services using notes, grew worries of replication of these notes. These worries can be evidently proved with some of the early inscriptions on the Chinese Notes which read "Those who are counterfeiting will be decapitated". Counterfeiting is the act of replicating currency notes without the authenticity of the state [2]. Counterfeiting usually occurs like a skeleton in a closet. With the access of modern-day technology to the general community, counterfeiting has become apprehensive. Counterfeiting has many ill-effects which include inflation of economy, reduction in the value of money & abate of acceptability. When a currency is replicated without any regulations, prices of goods, commodities & services surge exponentially, become exorbitant thereby erode the purchasing power. In today's world, every nation engages itself significantly in Trading (imports & exports of goods). Inflation leads to the collapse of the exchange rate (weaker currency scenario). Economies now would have to incur higher prices in local currency when its value tumbles down against the trading partner's currency. This creates hysteria among people in the financial world. Payees begin exploring & demand for substitute of currency, a transition thus evolves from currency to other forms of payment like digital currency, gold exchange etc., potentially creating a catastrophic loop since these novel forms of payment are less widespread. All the above-mentioned problems lead towards hitch & dissimilitude in the financial world. Hence, it is very essential to keep a check on Counterfeiting.

Many counterfeiting regulations measures have been developed & deployed by Note Minting Agencies in various nations. Some of the traditional Anti-Counterfeiting measures are raised intaglio printing on notes for easy recognition of forgeries, intrinsic & extrinsic study of the surface, Holograms, Fluorescence (Transmittance & Reflectance) analysis. Intaglio printing refers to the design being engraved into the surface. Intrinsic & Extrinsic study of the surface is performed to evaluate the legitimacy of the note based on the properties of the material which was used to print the note. Holograms are interference patterns when illuminated at a particular position form a 3-dimensional image. Based on Hologram formation, the legitimacy of notes is sometimes deduced. Fluorescence Check is carried out by passing light through the surface (note) & properties like transmittance & reflectance are noted. A threshold value is established, above which the note is classified to be legit, below which the note is regarded as fake.

Over time, currency notes are being replicated with profuse caution & care, perfectly resembling a legitimate currency note. Identification of original & fake notes has become more perplexing. Advancements in the areas of cloning, imaging, and colour-printing have continuously been integrated into reproducing fake notes, thus creating trouble for the economy. Hence, this work aims to engineer an efficient framework for evaluating the legitimacy of a bank currency note & classifying them as original & fake notes & propose a feasible solution that can confront the above-mentioned problem with ease.

It can be achieved through Machine Learning Classification Techniques. In this work, Random Forest & Naïve Bayes Classifiers have been employed & their results are compared. After comparison of results based on Accuracy, a

classifier is picked & a re-usable model is developed using Pickle. Data used for training the model was obtained from Kaggle. It was publicized by UCI in its ML Repository as a CSV file & was collected from images of both original & forged note specimens captured using an Industrial Camera. The size of each image used was 400 x 400 pixels. Wavelet Transformation was later applied to each image specimen to extract features (statistical values) including variance, skewness, kurtosis & entropy. This is a binary classification problem, so the last attribute in the dataset adopted for training is the class label of the corresponding record. After engineering a classification model, a well-structured UI has to be fostered for accessing of the model developed. UI has been developed by dint of Flask, a web-development framework & Flassger, a Swagger API. This paper is developed in Windows OS, but the irony is this solution would be used in financial & banking sectors which usually operate servers with Linux distributions as their OS. Ultimately, this paper is deployed into docker for environment standardization. Environment standardization speaks about the paper running decently & optimally irrespective of the platform it is executing on without installation of additional requirements.

Related Work

The authors [1] propose a mechanism for distinguishing between legitimate & forged notes. Experimentation focuses on the usage of Random Forest classifier and ratio in which training & test split should occur for achieving best results. Four Attributes were considered for training the model & they are (i) Variance, (ii) Skewness, (iii) Kurtosis & (iv) Entropy of the wavelet transformed images. Each of these features holds continuous values & output label i.e., Class of the banknote which is an integer & occurs to happen as a binary value. It can hold values either 0 (or) 1. Authors in their work had two Trains versus Test Splits of the dataset in the ratios 60:40 & 80:20 & later applied Random Forest Classification on the dataset. Random Forest is an ensemble learning technique that works by constructing elementary decision trees, hundred in numbers by default, which can be changed according to the need for making predictions. Decision trees in Random Forest construct themselves by iteratively partitioning themselves based on Gini Impurity (or) Information Gain & Entropy at each node. Here Gini Impurity, Information Gain & Entropy are heuristics which help to build decision trees to predict output class label more accurately.

$$\{H(x, \theta p), p = 1 \ldots n\}$$

Here, $H(x, \theta)$ is the Hypothesis, x is the feature matrix, p is a positive integer from 1 to n representing each decision tree & n is any positive integer. If, output class predicted by each decision tree is represented by $h(x, \theta 1), \ldots, h(x, \theta n)$, then the predictor of random forest classifier is the output class label which is predicted by the majority of elementary decision trees, represented as follows:

$$HRF = \text{majority vote } (h(x, \theta k))$$

Authors finally conclude that the 80:20 split outperforms the 60:40 split, gives a 95.3% success rate & is an efficient way to detect forged banknotes. The works elucidate the process of feature subset selection which is quite important as it helps identify significant attributes & improves the prediction of the model [2]. Boruta Package in R Programming language was applied on Chronic Kidney Dataset from UCI Machine Learning Repository. Features with less significance are disregarded in each iteration. Random Forest Classifier is employed on feature subset selection, classification & regression to examine its functioning from diverse perspectives. The authors state that Random Forest Classifier is very robust and operates efficiently even for a dataset with a higher number of input variables. Each split in the decision tree happens based on the purity of a node. A node is said to be pure if it results in entities of a single class. Here, Random Forest Package in R Programming language is used for classification & regression analysis. But they also express their concern about the error rate in Random Forest Classifier which surges with the rise in the correlation between attributes. They also bring to light a mechanism that the classifier employs while constructing of the forest. This mechanism calculates an unbiased error quantity of the forest implicitly, eliminating the need for cross-validation. This is accomplished by means of an out-of-bag (OOB) score.

$$OOB\ Score = k$$

Here k is the count of correctly predicted rows from the out-of-bag sample (data records omitted from dataset while considering bootstrap samples).

The Authors [3] explain HAAR Discrete wavelet transform for Image Compression. They describe wavelets as a waveform of limited duration with mean zero which are used for analyzing information with respect to frequency & time. They stress on image compression because images with smaller sizes can be easily stored & transmitted. Analysis of information in the image using Fourier Transform cannot help as such because there is no means of identifying exactly where an event occurs since it is a function of frequency & magnitude. In Discrete wavelet transform, all the wavelets are discrete. They demonstrate how HAAR Wavelet transform executes like any other wavelet transform. Wavelets are decomposed using Filter Banks (high pass & low pass filters). The signal is passed through High Pass Filter (HPF) and low pass filter (LPF) and down sampled by two to obtain High-Frequency components or detailed coefficients (through HPF) & low-frequency components or Approximation Coefficients (through LPF). When Multi-level decomposition happens at filter banks, we observe that some particular level of Approximation Coefficient, we see the removal of noise and get an approximation of the image. They conclude that HAAR Discrete Wavelet Transform can be used to gain promising quality images without compromising the details in the image after compression.

The Authors [4] make a prediction about a variety of diseases & the display of their results is projected through a user interface developed with the help of Flask. They explain how machine learning algorithms like SVM and Random Forest have been used to predict the class of the disease. Authors in their work, majorly emphasize how they have integrated the capabilities of machine learning with a friendly & facile interface. After processing the data with the training set & test set, the best algorithm is garnered & developed into a model by pickling it. Pickling is the process of serializing & de-serializing python object structures. When a model is built & pickled, it can be now easily integrated into Flask & called on-demand. This model can now be re-used & helps overcome the overhead of compilation of multi-disease prediction algorithm time &

again. Once, flask routes are specified, they can be verified from the front-end by initializing the flask debugging server.

The work [5] demonstrates the use of Docker, Kubernetes & Google Cloud Platform for continuous development & deployment into production. These cutting-edge technologies assist in constructing a variety of architecture anywhere. Dockers are used to develop, ship and execute solutions on any machine independent of the system configuration. It makes deployment simpler & faster. Docker Container Images are highly efficient, which means that the same image can be run across multiple client machines parallelly. Docker hub helps manage all the docker containers. The author also throws light on Kubernetes, which is a container- management software, scaling platform that provides resources on-demand & also monitors resources deployed. Google Cloud Platform is used to deploy containers in Kubernetes. This work also discusses how docker containerization is performed. The authors also explain how a content management system website like WordPress can be hosted in Docker.

METHODOLOGY

A. Dataset Elucidation

Dataset utilized for developing this paper was published in Machine Learning Repository by the University of California, Irvine. Later, it was made available on Kaggle with the title 'Bank Note Authentication UCI Data' from where it was imported for analysis & accomplishing of this work. Dataset published by UCI is in the form of CSV (comma-separated-values) of five attributes, which were obtained from multiple banknote images by performing a wide variety of operations. This data was acquired from images captured of both forged & genuine specimens. Operations that were performed to deduce the attributes in the CSV from banknote images are mentioned, as we move further through this section. An Industrial Camera was employed for capturing the illustrations available on the notes, digitization & print inspection of the banknotes. The size of each image is estimated to be 400 x 400 pixels. After acquiring each image sample, it is transformed into a grey-scale self. Grey-Scale images with resolution 660 DPI (Density Independent Pixels) are now gained. The final operation which is performed on grey-scale images is wavelet transformation to identify our Regions of Interest (ROI) and deduce information or features conveyed by ROI(s). Thereby, we procure CSV file & essentially the features – variance, skewness, kurtosis, entropy & ultimately the class label of the sample.

B. Feature Interpretation

Variance - Variance of a wavelet transformed image can be defined as alterations in pixel values due to challenges like the difference in viewing geometry of banknote image. Variance portrays the change brought because of geometrical differences in capturing Currency Images. This attribute holds continuous values. Skewness – In the world of Digital Processing of images, Skewness is a property by virtue of which we can arrive at conclusion about the quality of wavelet transformed image surfaces. Surfaces that are dark & lustrous are said to be more positively skewed in contrast to lighter & granular surfaces. Skewness explains whether the currency image captured has a darker or lighter surface. This attribute possesses continuous values. Kurtosis- Kurtosis of a wavelet transformed image is used to describe noise & resolution measurements in the digital processing of images. Kurtosis is inversely proportional to Noise & Resolution. Thus, we can infer that High Kurtosis refers to low noise & resolution, and vice-versa. Kurtosis for a banknote can suggest the percentage of noise & resolution applied. This is also an attribute holding continuous values. Entropy- Entropy of a wavelet transformed image can be defined as the dissimilarity in the energy composition across an image. Entropy can be calculated from a banknote since we can observe notable differences in the print imbibed on the note. This attribute too holds continuous values. Class attribute holds integer values & is a categorical attribute. It tells whether a record with the above features is legit or forged.

C. Pre-Processing, Analysis & Classification

Consider, any real-world dataset, it typically occurs in a large number of dimensions i.e., a large number of rows and columns. Pre-processing of the dataset is referred to as operations performed such that attributes can now be easily understood by the algorithm, thus can be easily parsed. Pre-processing techniques to be applied are generally selected based on the kind of data we are handling. In our work, one such pre-processing technique which needs to enforce is checking for missing values, imputing them if any. It is very likely to have missing values in our dataset which deviate the results from the expected, thus missing values should be checked & imputed by the mean value of that attribute or by any value which provides the best imputation. This technique falls under a broader category of Data Quality Assessment. It is completely unrealistic to expect that the data we obtain from the real world would be of high quality & ideal. Data Quality Assessment ensures that we provide good quality data to the algorithm for parsing. This work is developed in Jupyter Notebook using Python Programming Language. Missing value treatment is hence accomplished in Python using the following code:

df.isnull(), where df is a data-frame that consists of dataset imported

The dataset imported in this work, do not consist of missing values, therefore we locomote towards further steps. Once the dataset is clean and organized, we split the dataset into independent features & dependent features. In this paper, Variance, Skewness, Curtosis & Entropy are independent features whereas class is a dependent feature. This split can be achieved in the following line of code:

x = df.iloc[: , : -1], x is now data-frame comprising of independent features

y = df.iloc[: , -1], y is now data-frame comprising of dependent features

Now, Independent Features & Dependent features are split into Train & Test Data. [6] Training-Test Split is the procedure of taking a dataset & breaking them down into elementary subsets in a particular ratio. The first subset is used to identify the best fit model, therefore known as Train Dataset. The second subset is used to evaluate the performance of the model, by comparing the predicted values against the actual values, hence known as Test

Dataset. In this work, the train-test split ratio adopted is 70:30. It can be written in Python as follows:

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,random_state = 0)

Here, x is independent and y is dependent features, x_train & y_train are training sets and x_test & y_test are test sets. Random_state can be any constant value ∈ Integers.The problem being solved in this work is a classic example of Classification since the output label is going to be either 0 or 1 i.e., forged or genuine. Once Train & Test datasets are at disposal, a classifier can be employed for fitting a model on the data being training with. Classifiers are used to predict the class label of given data points. Numerous classifiers are available, but the classifier which is going to be employed depends on the problem being dealt with and also on the nature of the dataset. This work explains a comparison of two classifiers, employed on the dataset and which are Random Forest & Naïve Bayes.

Random Forest compromises a horde of individual decision trees. It makes use of bagging and feature randomness for building an uncorrelated forest of decision trees. Bagging can be demonstrated with the following example if the training dataset consists of N Samples, e.g. [1, 2, 3, 4], then one of the inputs which might be given to trees would be [1, 2, 3, 2]. Both the above lists are of the same length & elements in the latter input list consists of a few numbers which are randomly selected & replaced with other elements. Through this, we can understand that bagging is sampling with replacement. Feature Randomness can be explained by understanding that for building a decision tree, each possible feature is considered & one giving us a purer split is considered. Examples of Pure Split is (9 Positives, 1 Negative), (8 Negatives, 2 Positives) and Impure Split are (5 Positives, 5 Negatives). Decision trees are formed by recursively considering each record of each attribute. They operate on a principle that the feature selected for splitting should help us reach class labels at lower depths. For achieving this, for every split in Decision Tree Entropy, Gini Impurity and Information gain for that feature are calculated. Entropy defines the measure of the purity of a split. Entropy ranges between 0 to 1. 1 as entropy indicates that it is a completely impure split & 0 says that it is a pure split. Entropy is represented as:

$$H(S) = -P_+ log_2(P_+) - P_- log_2(P_-)$$

Here H(S) is Entropy of S which is a subset of Training Example, P_+ & P_- are Percentage of Positive & Negative classes.Information gain is the amount of reduction in entropy or randomness after each consecutive split. It is represented as:

$$Gain(S, A) = H(S) - \sum \frac{|S_v|}{|S|} H(S_v)$$

Here S is the subset of Training Example, A is an attribute for which Information Gain is calculated, Sv is Subset after splitting, $H(S_v)$ is Entropy of subset after splitting. Feature splitting structure with the highest Information gain will be selected and implemented for the construction of the Decision Tree. Gini Impurity is very similar to Entropy and serves the same purpose as does Entropy, but Gini Impurity is more computationally efficient than Entropy because it doesn't include logarithms. It is represented as:

$$GI = 1 - \sum_{i=1}^{n} (P)^2 = 1 - [(P_+)^2 + (P_-)^2]$$

Here GI is the Gini Impurity for an attribute and P_+ & P_- are Percentage of Positive & Negative classes.

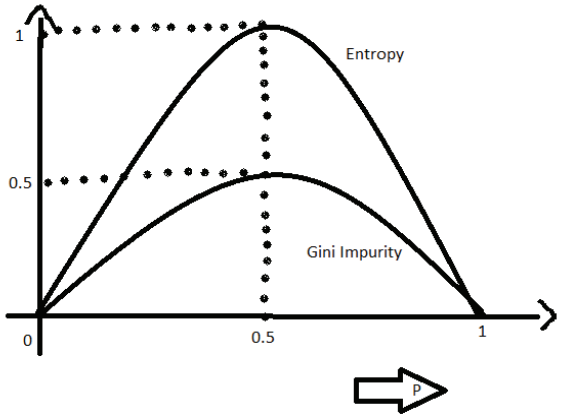

Fig. 1. Gini Impurity vs Entropy

This can be penned down in Python as, from sklearn. ensemble import RandomForestClassifier, from sklearn import tree, classifierRF = RandomForestClassifier(), lassifierRF.fit(x_train,y_train)

By default, hundreds of Decision Trees will be constructed for a Random Forest. The number of estimators can be changed. A particular decision tree out of these hundred can be visualized. It is possible in the following way:plt.figure(figsize=(15,10)),tree.plot_tree(classifierRF.estimators_[2], filled = True) as mentioned in fig.2.

Prediction of Random Forest will be the output label which majority of trees in the forest predict & this prediction will be more accurate than any elementary decision tree in the forest. It is done in the following way:

y_predRF = classifierRF.predict(x_test),

Later accuracy for Random Forest is also calculated, a confusion Matrix is also deduced & it is written in Python as:

from sklearn.metrics import confusion_matrix, accuracy score , scoreRF = accuracy_score(y_test, y_predRF)

cmRF = confusion_matrix(y_test, y_predRF)

print('Random Forest Accuracy : ',scoreRF)

print('Random Forest Confusion Matrix :',cmRF)

Through Experimentation of Train – Test Dataset Split in the ratio 70:30, for the dataset which we have imported, on the application of Random Forest Classifier, we receive an accuracy of 98%.Naïve Bayes can also be applied on the dataset to fit a model for predicting the results. [7] It is a classifier based on conditional Probability adopted from Bayes Theorem with an assumption that attributes are conditionally independent. [8] Bayes Theorem helps us calculate maximum posterior probability P(C | X), which means Probability of Happening of an event C on occurrence of Event X.

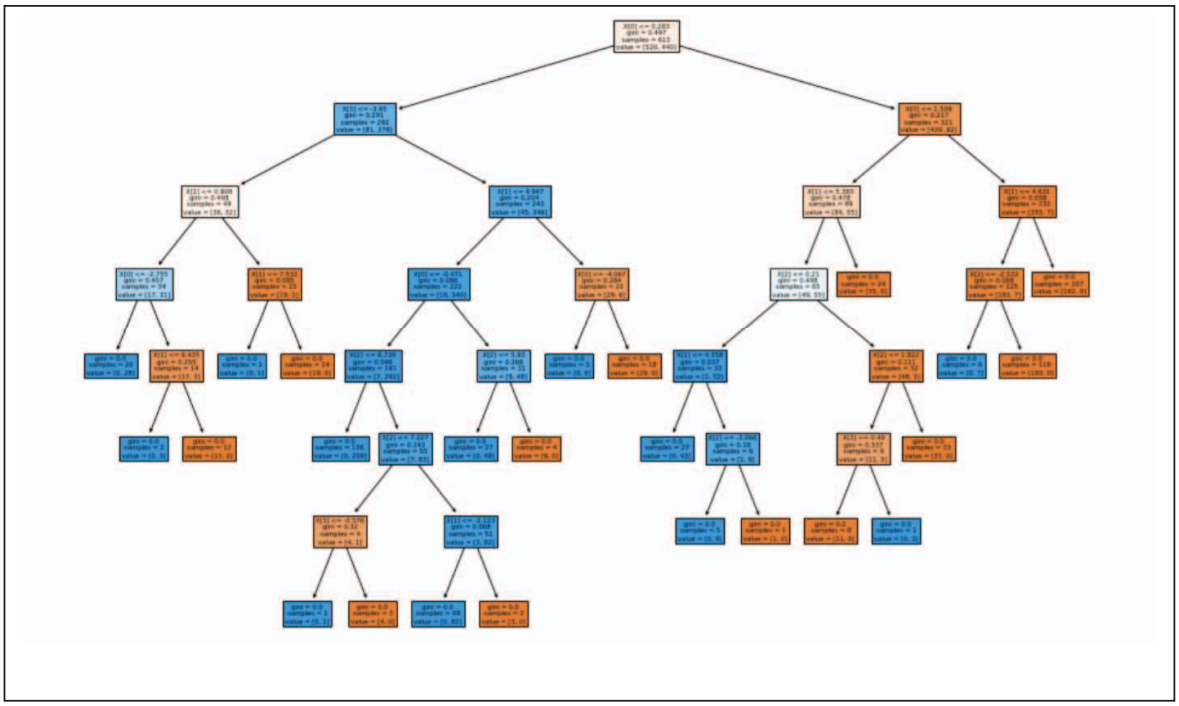

Fig.2 Decision Tree 2 in Random Forest

$$P(C \mid X) = \frac{P(X \mid C)\,P(C)}{P(X)}$$

Here P(C | X) is posterior Probability of a class, P(c) is the prior probability of class. P(X | C) is probability of predictor in given class & P(X) is the prior probability of predictor.

Through Experimentation of Train – Test Dataset Split in the ratio 70:30, for the dataset which we have imported, on application of Naïve Bayes Classifier, we receive an accuracy of 83%.This work uses Random Forest Classifier for developing an application to classify the legitimate and fake currency note specimens. After modelling an algorithm for classification, we will pickle this model for re-usability. Pickling of a model is defined as serializing & deserializing a python object structure. Any object that is pickled can be saved on disk & used later. This is achieved in python in the following way: import pickle,

pickle_out = open("classifierRF.pkl","wb"),

pickle.dump(classifierRF,pickle_out), pickle_out.close()

D. User-Interface with Flassger, A Swagger Api

This pickle file can now be integrated into any framework like Flask, for developing user interface for checking the results. Flask is a web-development micro-framework for developing the user interfaces with Python. This work uses Flassger, a swagger API for erecting an interface. Flassger API helps to add fields and create an interface easily with less code. Backend logic of sending the values entered by the user to the Pickle File is handled by code written in Python in Flask. To make the predictions in the user interface, the Pickle file is to be loaded in the Flask & will be opening in read-write mode. We also need to specify Routes to load the page on specifying the URL in a web browser. After specifying the routes, we'll be giving description for auto-generation of the user-interface without any Python Code using Flassger. Flassger is very particular about the Indentation of code & may fail if not properly indented. Description and standard format for generating fields in user-interface without python & HTML Code using Flassger is as follows: " " " Title of the Interface

parameters: - name : parameter_name

in : input type of variable, can be either query or form etc

type : type of variable, can be a number or string etc

required : makes the field mandate, can have either true or false reponses: response_status_number : description : response_message

In this work, we require four variables so in Flassger, we create four parameters namely variance, skewness, curtosis & entropy. Next, we put another parameter response which catches the response status we receive from the server and displays the description associated with the response status. When we get response status as successful then our output is displayed. The Potential of Flassger lies in its ability to create a well-structured user interface with ease without actually writing long lines of code. Demo Code Snippet for Parameters of Flassger in Flask Code is: Let's authenticate the Bank Notes. This is using docstrings for specifications. parameters :- name : variance, in : query and type : number etc.

Once, the user interface is designed, this web application on running in Spyder triggers the server and the web page loads from Swagger Server & runs on localhost in our machine. Through this web page, on entering parameters, we get the class label for the record of banknote specimen. This entire application runs on localhost & can be accessed through the URL: http://localhost:8000/apidocs.

IV. Deployment in Docker

Since this application runs on localhost, so it can't be accessed on other devices without prior installation of packages & the required files for execution of the application. To make the application scalable, our API should be able to run on multiple devices in the production irrespective of their configurations & environment. Thus, Docker can be used for deploying Counterfeit Regulation Application. Docker helps us fix the problem of the environment, by standardizing the environment. Consider two machines one at the developer's site & another on Quality Assurance (QA) site with drastic changes in the operating system, the configuration of machines. Even though environments are different at the developers end & QA's end, once a docker container is created for a web application or a machine learning application, it will be built once & can be deployed anywhere. Docker is a part of DevOps which helps developers put together all parts & supporting files of an application in a single entity & ship it as a container. Docker containers are created on the top of the Operating System, each container has its unique process ID, network configuration consisting of port number & most importantly root folder comprising of the entire directory in the container. Docker also eradicates the problem of Isolation i.e., if three docker containers are created on the top of the operating system and one container fails, then the resources allocated to it are flushed & allocated to the other two, ensuring safe & smooth running of the containers. Docker also ensures Portability which means we can switch & place this container in any environment & still it runs in the same way. Some of the important commands in Docker are:

FROM, COPY, EXPOSE, WORKDIR, RUN & CMD.

Using these commands, docker images will be created in a virtual toolbox, i.e., Docker Desktop Application. These commands will be written in a Docker File. To start with Docker, a base image should be created which is to be downloaded from the docker hub, where all base images are present. Base Image is typically the operating system of the version which is needed. FROM command is used to import a base image from docker hub. COPY command is used to create a replica of the folder structure of the application in the Host Machine to root folder in the Docker Image. Docker Image is the encapsulation of base image, an environment, a server for execution of application & finally the application. Each Docker Image has a network interface that consists of port. If we expose a port in the Docker Image, then the complete Counterfeit Regulation can be accessed through this port number. EXPOSE command is used to expose the port number of the Docker Image. WORKDIR command is used to specify the path from where the application should start running. RUN command is used to install all the libraries & packages from the requirements.txt file that has already been copied to the root folder of Docker Image using COPY command. CMD command is used to initiate the execution of the application.

Here, Continuumio is the base image with bootstrapped installed Anaconda available in it. requirements.txt is the file containing all the libraries & packages that should be installed. To build the Docker Image, the command used as docker build -t ban_note_api. Here ban_note_api is the name of the Counterfeit Regulation Docker Container which is built once & can be shared anytime.

Once, the Docker Image is created, we dockerize it and deploy the application in the production environment. This Docker Image can be shared to Docker Hub from where others can pull the Docker Image. Command for pushing Counterfeit Regulation Docker Container to Docker Hub: docker push saikoushik00/ban_note_api

After pushing Docker Image to docker hub, anybody can install Docker Desktop in a device and pull this image using the following command: docker pull saikoushik00/ban_note_api .Once we pull the docker image in the secondary device, we should run the docker image to utilize the application. Command to run the Application in Docker Image is: docker run -p 8000:8000 ban_note_api

Eventually, Counterfeit Regulation Application can be accessed in secondary device through this URL: http://localhost:8000/apidocs.Ultimately, Counterfeit Regulation Application executes on the secondary device without installation of additional libraries on secondary device.

V. Result and Discussion

This work compares the performance of two classifiers: Random Forest & Naïve Bayes on the Bank Note Authentication dataset published in the Machine Learning Repository by UCI & made public on Kaggle. Accuracy as a parameter was considered to evaluate the performance of each classifier. This application is a problem of binary classification. Class Label 1 refers to Genuine Currency Note & Class Label 0 refers to a Replicated Currency Note or a Forged Note. It was found that on application of Random Forest Classifier, on a dataset with Train-Test Split Ratio 70:30, resulted in an accuracy of 98%. Confusion Matrix for Random Forest Classifier is: [[229 3] [2 178]], which means on the Test Dataset, Classifier predicted Two hundred and twenty-nine samples as True Positives, i.e., both Predicted & Actual are 1, Three samples as False Positives, i.e., Actual is 0 (Fake), but is predicted as 1,Two samples as False Negative, i.e., Actual is 1, but predicted as 0 & One hundred and seventy-eight samples as True Negatives, i.e., both Predicted & Actual are 0.

Naïve Bayes Classifier was later enforced on the same dataset with Train-Test Split Ratio 70:30, resulting in an accuracy of 83%. Confusion Matrix for Naïve Bayes Classifier is: [[199 3] [36 144]], which means on the Test Dataset, Classifier predicted One hundred and ninety-nine samples as True Positives, i.e., both Predicted & Actual are 1, Three samples as False Positives, i.e., Actual is 0 (Fake), but is predicted as 1,Thirty six sample as False Negative, i.e., Actual is 1, but predicted as 0 & One hundred and forty-four samples as True Negatives, i.e., both Predicted & Actual are 0. These results proclaim that Random Forest Classifier on Bank Note Authentication dataset outperforms. Naïve Bayes Classifier & thus, is adopted for modelling the algorithm for the classification of original & fake bank Note. This work is an application majorly used in Banking Sector.

Most of the banking sector applications run on servers with Linux operating systems.

TABLE.1. CLASSIFICATION REPORT FOR NAIVE BAYES MODEL

	precision	recall	f1-score	support
0	0.85	0.86	0.85	232
1	0.81	0.80	0.81	180
accuracy			0.83	412
macro avg	0.83	0.83	0.83	412
weighted avg	0.83	0.83	0.83	412

TABLE.2. CLASSIFICATION REPORT FOR RANDOM FOREST MODEL

	precision	recall	f1-score	support
0	1.00	0.99	0.99	232
1	0.98	0.99	0.99	180
accuracy			0.99	412
macro avg	0.99	0.99	0.99	412
weighted avg	0.99	0.99	0.99	412

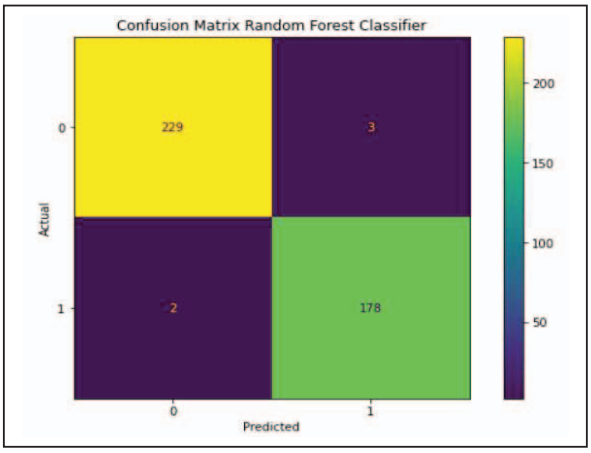

Fig.3. Confusion Matrix for Random Forest Classifier

But the authentication application which is developed in this work is built in Windows Machine. Docker is used for deploying the Counterfeit Regulation with Machine Learning Application in the Production Environment. This is done to standardize the environment where the application will be used.

VI.CONCLUSION AND FUTURE ENHANCEMENT

We live in a world where new technologies emerge every day & are accessible to people at low prices very soon. With this available technology, people have begun exploiting it and replicating the currency without the authenticity of the Money Regulating Body in a nation. This can harm the economy of the nation & should be regulatory measures should be implemented. On careful observation of various traditional techniques, scientific technologies to evaluate the legitimacy of a currency note, this work proposes a cheaper

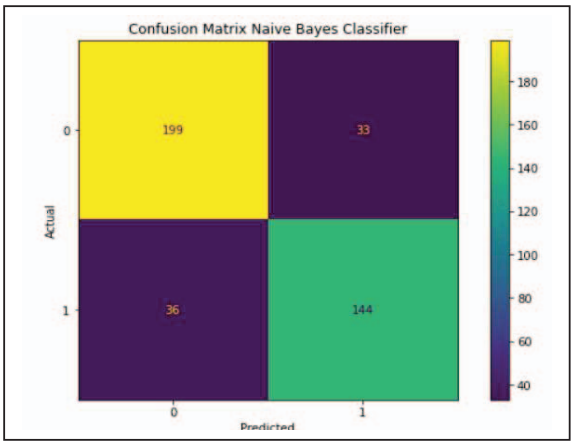

Fig.5. Confusion Matrix for Random Forest Classifier

	Classifiers	
	Random Forest	**Naïve Bayes**
Accuracy	98%	83%

Fig.4. Accuracy Comparison of Classifiers

and viable solution for classifying a forged & genuine banknote. This work relies on Random Forest for classification. This work can be extended by employing Neural Network for the identification of more prevalent features from the banknote & classify in a more efficiently

REFERENCES

[1] Rishabh Jaiswal and Suhani Jaiswal, "Bank Note Authentication using Random Forest Classifier", Volume 04, Issue 4, October 2019, Journals for International Shodh in Engineering and Technology.

[2] J. K. Jaiswal and R. Samikannu, "Application of Random Forest Algorithm on Feature Subset Selection and Classification and Regression," 2017 World Congress on Computing and Communication Technologies (WCCCT), 2017, pp. 65-68, doi:10.1109/WCCCT.2016.25.

[3] H. Kanagaraj and V. Muneeswaran, "Image compression using HAAR discrete wavelet transform," 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), 2020, pp. 271-274, doi: 10.1109/ICDCS48716.2020.243596.

[4] A. Yaganteeswarudu, "Multi Disease Prediction Model by using Machine Learning and Flask API," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1242-1246, doi:10.1109/ICCES48766.2020.9137896.

[5] J. Shah and D. Dubaria, "Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0184-0189, doi:10.1109/CCWC.2019.8666479.

[6] S. V. Patel and V. N. Jokhakar, "A random forest based machine learning approach for mild steel defect diagnosis," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2016, pp. 1-8, doi:10.1109/ICCIC.2016.7919549.

[7] J. Xu, X. Mei, X. Wang, Y. Fu, Y. Zhao and J. Wang, "A Relative State of Health Estimation Method Based on Wavelet Analysis for Lithium-Ion Battery Cells," in IEEE Transactions on Industrial Electronics, doi: 10.1109/TIE.2020.3001836.

[8] A. Anuragi, D. S. Sisodia and R. B. Pachori, "Automated Alcoholism Detection Using Fourier-Bessel Series Expansion Based Empirical Wavelet Transform," in IEEE Sensors Journal, vol. 20, no. 9, pp. 4914-4924, 1 May1, 2020, doi:10.1109/JSEN.2020.2966766.

[9] P. Vogel, T. Klooster, V. Andrikopoulos and M. Lungu, "A Low-Effort Analytics Platform for Visualizing Evolving Flask - Based Python Web Services," 2017 IEEE Working Conference on Software

Visualization (VISSOFT), 2017, pp. 109-113, doi: 10.1109/VISSOFT.2017.13.