BANK NOTE AUTHENTICATION USING ML

Project report submitted in partial fulfillment of the requirement for the degree of Bachelorof Technology

In

Computer Science and EngineeringBy

Vishal (181313)

UNDER THE SUPERVISION OF

Dr. Shubham Goel

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,Waknaghat, 173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"** Bank Note authentication using machine learning**"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2022 to May 2022 under the supervision of **Dr. Shubham Goel,** Assistant Professor Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Vishal (181313)

This is to certify that the above statement made by the candidate is true to the best of my knowledge

Dr. Shubham Goel

Assistant Professer

Computer Science & Engineering and Information Technology Jaypee University of

Information Technology, Waknaghat,

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makesus possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Dr. Shubham Goel, Assistant Professer** Department of CSE Jaypee University of Information Technology,Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of **data science and Machine learning** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Shubham Goel,** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my

parents.

Vishal (181313)

TABLE OF CONTENTS

Content **Page No.**

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVM: Support Vector Machine
API: Application Programming Interface
KNN: K-Nearest neighbor
DT: Decision Tree

ABSTRACT

This machine learning web application uses random forest classifications with 92% accuracy to assess the authenticity of original or counterfeit notes, along with many other note features such as variation, curvature, kurtosis and entropy. This dataset is sourced from the UCI repository and is also available on Kaggle, which captures data from images taken from models such as the original and the counterfeit notes. For digitization, commonly used industrial cameras were used for print inspection.

The final images are 400x400 pixels. Due to the distance from the object lens and the object under observation, gray-scale images with a resolution of approximately 660 dpi were obtained. Violet conversion tool is used to capture the feature from the image, in this ML model we see 1 for real note and 0 for duplicate note. Accuracy of the algorithms has been compared using various factors such as f1 score, precision, recall. Model has been saved in pickle file format so that it can be used.

Flask module has been used at the backend. Web API has been build. Full pipeline has been created for different tasks such as data preprocessing, hyperparameter tuning, training phase and testing phase.

Also Streamlit has been used to build frontend web API.

Chapter 1 INTRODUCTION

1.1 Introduction

As we know in the Global cash transactions still very important as there is increase in use of electronic transactions and decrease in the use of currency . To keep on financial activities banknotes are used. To continue smoothly, cash transactions, entry of counterfeit notes into circulation should be prevented. In the market there is continuous increase of fake notes. Whose copy is fake money? Notes are made illegally for various purposes. These fake notes are made in all denominations which lowers the country's financial market. Various advancements in scanner and copy machines have inspired crooks to make copies of banknote. To find out the fake note it is very-very difficult for human. Banknotes should have security aspects considered and security features to be introduced to reduce counterfeit currency. That's why it is desperately needed to make a system which find a note is real or not real in banks and ATM machines.

1.2 Problem Statement

Building a classification methodology to detect if a banknote is real or fake. It is a binary classification problem.

The following sentences provide a list of variables in the dataset:

- Kurtosis (continuous).
- Variance (continuous).
- Entropy of image (continuous)
- Skewness (continuous).
- class - integer.

1.3 Objectives

As everyone knows that banknotes are the very important assests of our country. In the market of financial some crooks present forged note on which similar to real note to make discrepancies of money. It's hard for humans to tell Genuine and counterfeit banknotes are different, because there are many features that are similar. Counterfeit notes are made with precision, so that’s why it is so important that there must be good algorithm which can find the bank note is genuine or fake. So the objective of this project to find the authentication of bank notes.

1.4 Methodology

There are different type of Machine Learning algorithms such as Supervised, Unsupervised and Reinforcement learning. Our problem is a classification problem which comes under supervised machine learning. There are various types of machine learning algorithms such as XGBoost, Random Forest, K Nearest Neighborhood, AdaBoost, Decision Tree etc. But in this project, we are using Random Forest classifier and K Nearest Neighborhood to build a methodology to predict whether a bank note is genuine or not. We also performed exploratory data analysis. Also removed skewness in the data.

1.5 Organization

At this stage, the structure of the chapter and chapter report report. We have previously seen the Introduction and Problem Statement in Chapter 1. Next, we look at the objectives and methods of the report. We are now part of the Reporting Society, which will clarify the chapters and topics discussed below. A brief description of the audit process will be presented in Chapter 2. This will include some of the resources we used in the past to obtain the necessary information for this project, as well as some of the work we have done through the community that helped us find it. until we were able to complete the task and lay the foundation for future solutions to similar problems. We will then move on to Chapter 3, where we will talk about how our model conceived and developed, First, we will go through the database we used to create this report and look at some of its key features. We will pass through a basic structure that leads to a random forest path, which includes decision trees and selection features. We will compare the various models we have used with the random forest and their comparative accuracy. We have gone through them briefly to help the reader understand them better. Following that, we have included a list of other mathematical formulas used in this study.

In Chapter 4, there is a demonstration of how the system works and compares it with other systems such as the Random Forest, KNN, and others. Then, at various stages of project work, I begin to show results, which include multiple graphs and statistics. This will be followed by the final result of the model.

In addition, in Chapter 5, we started with the concluding paragraph by discussing what we have achieved in the project and how this model was a good way for us to read and evaluate this topic. After that, we have introduced some future plans that can be developed for this project and added to it

Chapter 2 LITERATURE SURVEY

One of the most pressing concerns is the preservation of the reality of printed high-denomination banknotes. Every country's financial activity are heavily reliant on bank notes[1]. The study in[1]evaluates several machine learning algorithms and concludes that Decision- Tree and MLP Techniques are the most effective for classifying bank notes. The banknote's features are eliminated in [2] using a rapid wavelet transform. A categorization method has been devised that divides the note into four categories: real, high-quality counterfeit, low-quality counterfeit, and low-quality counterfeit, yielding an unreasonably high ROI with a 100% detection rate. SVM and BPN are compared in [3] and [4], with BPN outperforming SVM. [3] shows that BPN has a 10% greater accuracy than SVM and is faster Although BPN performance in SVM is stronger in [5] and [6, the best classification to estimate protein sequence is based on their compositions. The outcome is determined on the dataset type and classification complexity.

[7] shows how to use neural networks to classify Thai banknotes. To begin, a scanner captures images of notes, which are then recorded as bitmap data. BPN is used to learn and recognise features taken from this data. A new method for banknote recognition is proposed in [8], which uses a Probabilistic Neural Network (PNN) and achieves a 100% success rate. For banknote recognition, [9] employs the LVQ classifier. The experiment was carried out using US dollars, but it may be extended to other banknotes as well.

A technique for detecting counterfeit cash banknotes is presented in the paper [10]. The note is processed by a camera, and the image is segmented into sections. To match the watermark with Gandhi's face, histogram characteristics from distinct segmentation regions are derived. The outcome is displayed on the user interface. The features are derived from photos by segmenting the image in a similar work [11]. These characteristics are fed into an SVM classifier, which determines the note's genuineness.

In [12], a mechanism for recognising euro banknotes was proposed. The study found that using banknote photos as inputs, a three-layered perceptron can reliably classify banknotes into a certain class. The back-propagation approach was used to train this model.

Chapter 3 SYSTEM DEVELOPMENT

3.1 Analysis/ Design/Development/Algorithm

Data Set Used in the Minor Project

Dataset is taken from **UCI repository.** UCI is the great source of ML datasets. UCI repository containsthe domain theories and data generators and alsodatabases that mostly used in ML algorithms.

Link:https://www.kaggle.com/ritesaluja/bank-note-authentication-uci-data

3.2 Algorithms/ Pseudo code of the Project Problem

3.2.1 XGBOOST:

XGBoost (*extreme gradient boosting*) regularizes data better than normal gradient boosted Trees.It was developed by Tianqi Chen in C++ but now has interfaces for Python, R, Julia.

The objective function of XGBoost is the total loss function evaluated on all predictors and the regularization function (*j* tree) for all predictors. Meaning in principle, the estimate coming from the tree.
$(\theta)=\sum(yi-yi\hat{})+\sum\Omega(fj)\mathrm{obj}(\theta)$

Dependency of loss function is on the task being performed (classification, regression, etc.) and a regularization term is described by the following equation:

$\Omega(f)=\gamma T+1/2(\lambda\sum w^2 j)$

First part (γT) is responsible for controlling the overall number of created leaves, and the second term ($1/2\lambda\sum^{T} j=1\ (w^2 j)$) watches over the scores.

Mathematics Involved: Unlike other tree-buildng *i* algorithms, XGBoost does not use entropy or Gini indices. Instead, it utilizes gradient (the error term) and hessian to construct the trees. Hessian fora Regression problem is the number of residuals and for a classification problem.

$$h_m(x) = \frac{\partial^2 L(Y, f(x))}{\partial f(x)^2}_{f(x)=f^{(m-1)}(x).}$$

L=Loss function.

- Initialize the tree with only one leaf.
- compute the similarity using the formula

  $Similarity=Gradient^2/hessian+\lambda$

  Where λ is the regularization term.

- Now for splitting data into a tree form, calculate

  $Gain=leftsimilarity+rightsimilarity-similarityforroot$

- For tree pruning, the parameter γ is used. The algorithm starts from the lowest level of the treeand then starts pruning based on the value of γ.

  If $Gain-\gamma<0$, remove that branch. Else, keep the branch

- Learning is done using the equation

  $NewValue=oldValue+\eta*prediction$

  where η is the learning rate

3.2.2 Random Forest:

The Decision Tree is one of the least biassed models, but it also has the greatest differences. However, the data is deleted by the decision trees. As a result, the bagging strategy would be an excellent way to lessen the decision tree difference. We may use the Random Forest instead of the bagging model with the underlying model as the decision tree, which is more convenient and well-optimized for decision trees. The fundamental issue with bagging is the lack of independence in the sample dataset. The benefit of random forest over the bagging model is that it alters the random forest bagging model's working algorithm to reduce tree correlation. When constructing a tree, the goal is to incorporate more coincidence to lessen correlation.

1) Bootstrapping is used to acquire distinct samples from the training dataset, similar to bagging.

2) We train our tree model on each specimen and increase the depth of the trees. The way the trees are generated in the random forest is different. We partition nodes using all sample data in bootstrapping, but not random forests. Each time a partition is created in a decision tree, a random pattern of ‘m' predictors is picked from the total 'p' predictors. For partitioning, only those 'M' predicates are employed.
Assume that one of the 'p' predictors is extremely powerful. This predictor is now present in every model. As a result, whenever trees for this sample data are constructed, this predictor is chosen to be split by all trees, resulting in the identical tree for each bootstrap model. This incorporates correlation into the dataset without causing a significant change in the average correlation dataset findings. As a result, the number of nodes available for division in a random forest is restricted, and the tree structure is similarly random.
The majority of prophets are not thought to be divisive. The value of'm' is usually calculated as m p, where p is the number of predictors in the sample. The random forest pattern appears when m = p.
the random forest pattern

becomes the bagging model. This method is also known as "feature sampling"

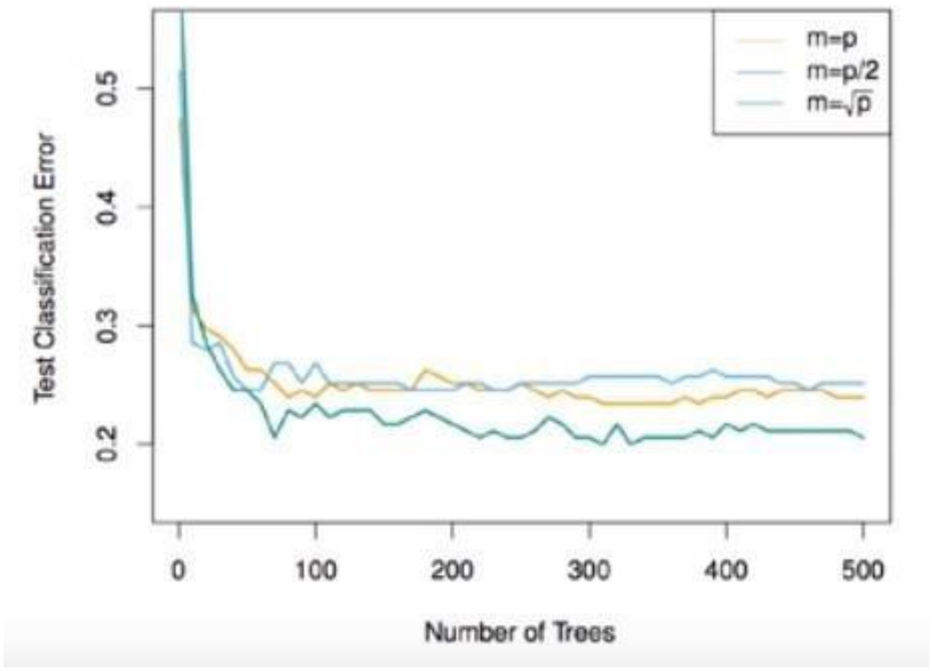


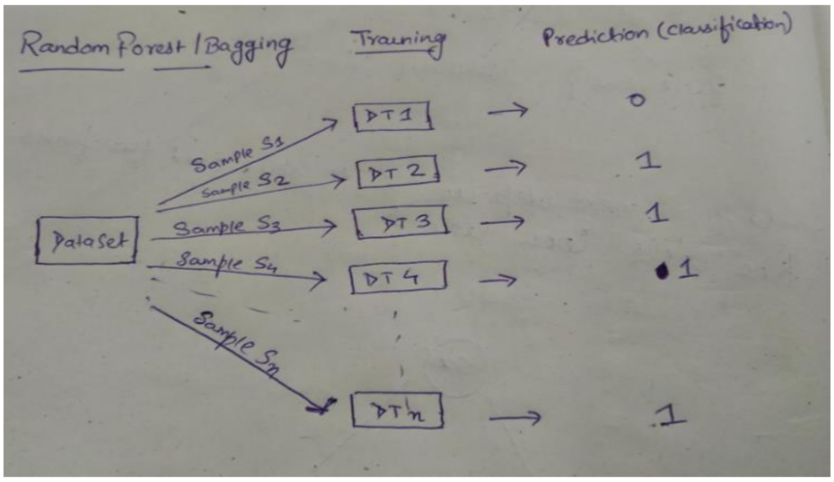
Figure 1 Random forest : classification Error V/S Number of Trees

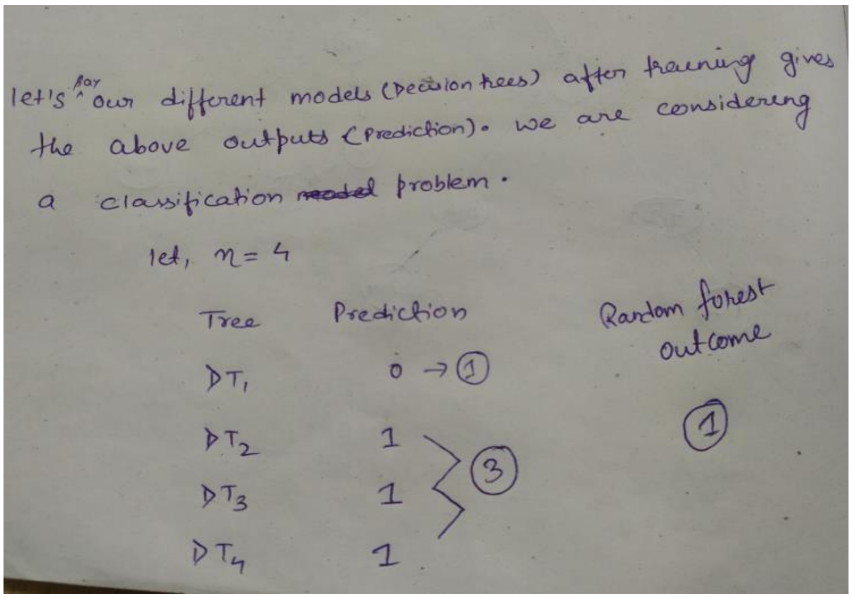
The above graph represents the decrease in test classification error as we select different values of‘m’.

(3) Once the trees are formed, prediction is made by the random forest by aggregating the predictions of all the model. For regression model, the mean of all the predictions is the final prediction and for classification mode, the mode of all the predictions is considered the final predictions.

Working of a Random Forest Model:

From the given dataset different samples are created by bootstrapping and these samples are used to train different decision trees. Once the training is complete, prediction is made using all the different models.





let's for our different models (Decision trees) after training gives the above outputs (Prediction). we are considering a classification ~~model~~ problem.

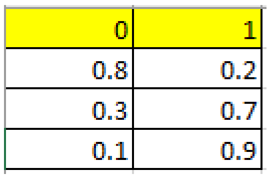
let, $n = 4$

Tree	Prediction	Random forest outcome
DT_1	$0 \rightarrow \textcircled{1}$	$\textcircled{1}$
DT_2	1	
DT_3	1 $\Big\}\ \textcircled{3}$	
DT_4	1	

Figure 2 Random Forest : Bagging

Predicting Outcome

Random forest makes the prediction by taking the mode of all the predictions made by all the models, since this is the case of classification. This process is also known as "Majority voting”. We can also use prediction probability to make the final prediction. Now we will use predict_proba method, its range is from 0 to 1 . If the classification problem is of binary , matrix containing 2 columns and various rows will be returned. Result will be :



0	1
0.8	0.2
0.3	0.7
0.1	0.9

Figure 3 Predicting Probability

Each row corresponds to an estimate. The first column is the probability that prediction is 0 and the second column is the probability that it will be prediction 1. Each row adds up to 1.Let us take only

second column, which will give the average value that will be predicted for that particular row. We will use continuous values as such instead of rounding them to 0 or 1 .Thus giving us a single vector. To get the mean we will add all the vectors and divide by total number of vectors to get the final value.Using predict_proba value will be rounded of to 0 or 1. Similar thing will happen in case of regression problem.

3.2.3 K Nearest Neighbor:

K-nearest neighbor is a supervised learning algorithm. Therefore, prediction requires both independent data as well as dependent data. This algorithm can be used for both regression problem as well as classification problems, but mostly it is used for the latter. Given a dataset with different classes, this algorithm makes a prediction by computing the Euclidean or Manhattan distance *i* between the test set and each training set. It then selects the k points that are closest to the data in the test set. Once the points are chosen, the algorithm calculates the probability (in the case of classification) of the test point belonging to k classes of training points and the class with the highest probability is chosen.Let’s understand this with an illustration:

1) Below training dataset that is already given . We have a new test data that we need to assign to oneof the two classes.

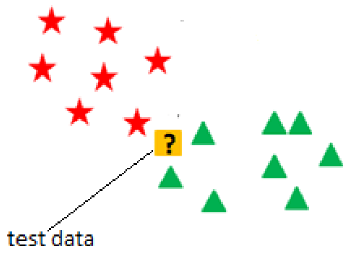


Figure 4: KNN unlabeled test dataset

2) Now, the k-NN algorithm evaluate the Euclidean distance between the data in test set and data intraining set.

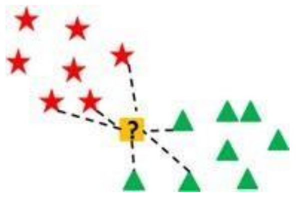


Figure 5 KNN : Clustering

3) After calculating the distance, it will choose the k training points which are nearest to the data(test).Let’s assume the value of k is 3 for our example.

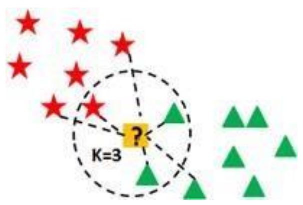


Figure 6 KNN Centeroid

4) Now, 3 nearest neighbors are selected, as shown in the figure above. Let’s see in which class ourtest data will be assigned :

Number of Green class values = 2Number of Red class values = 1 Probability(Green) = 2/3

Probability(Red) = 1/3

because the probability for Green class is higher as compare to Red, the k-NN algorithm will assignthe test data to the Green class.
Similarly, if this were the case of a regression problem, the predicted value for the test data will simplybe the mean of all the 3 nearest values.

This is the basic working algorithm for k-NN.

Euclidean Distance:

It is the most commonly used method to calculate the distance between two points. The Euclideandistance' is calculated as:

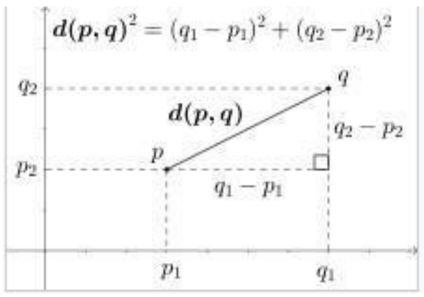


Figure 7: Euclidean Distance

Similarly, for n-dimensional space, the Euclidean distance is given as:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

Hamming distance

Used to calculate the number of mismatching points between two vectors.

$$HamD(x,y) = \sum_{i=1}^{n} 1_{x_i \neq y_i}$$

Generally, if we have features as categorical data then we consider the difference to be 0 if both the values are the same and the difference is 1 if both the values are different.

Manhattan Distance

Used to calculate the mode of differences between the different values in vectors.

$$MD(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

This is less influenced by outliers than the Euclidean distance. With very high dimensional data it is more preferred.

Lazy Learners

Lazy learners are a term that describes k-NN algorithms. Let's look at why that is. Eager learners refer to a variety of techniques such as Bayesian classification, logistic regression, and SVM. These methods generalise over the training set before obtaining the test data, that is, they build a model based on the training data before receiving the test data, and then predict/classify the test data. With the k-NN method, however, this is not the case. It waits for the test data before creating a generalised model for the training set. It begins generalising the training data to categorised the test data after receiving the test data. As a result, a lazy learner saves training data while waiting for the test set. Classification takes significantly more time than training for such algorithms.

.

Weighted Nearest Neighbor

We apply weights towards the k nearest neighbours in weighted kNN. The weights are mainly allocated based on the distance traveled. The rest of the data values are sometimes given a weight of 0. The basic idea is that neighboring points should be given greater weight than father points

Choosing the value of k

K has a huge impact on the k-NN classifier. With increasing 'k,' the model's flexibility reduces. When the value of 'k' is low, variation is large and bias is low, but as the value of 'k' increases, variance decreases and bias increases. With extraordinarily low values of 'k,' the method may overfit the data, whereas with very high values of 'k,' the algorithm may underfit the data. Check out the following trade-off between '1/k,' train error rate, and test error rate:

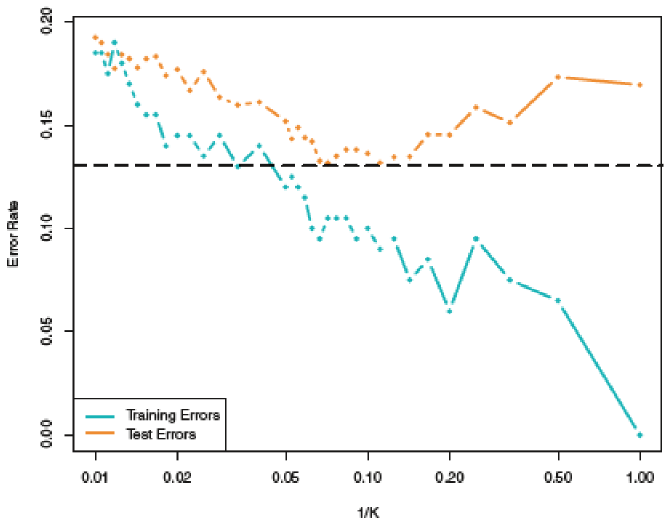


Figure 8: Weighted Nearest Neighbor

We can clearly see that the train error rate increases with the increase in the value of 'k' whereas test error rate decreases initially and then increases again. So, our goal should be to choose such value of 'k' for which we get a minimum of both the errors and avoid overfitting as well as underfitting. We use different ways to calculate the optimum value of 'k' such as cross validation, error versus k curve, checking accuracy for each value of 'k' etc.

3.2.4 Logistic regression:

This type of mathematical analysis is often used for predictable analysis and modeling, it is also called logit regression. In this analysis method, the dependent variations are limited or categorized: either X or Y (binary regression) or can say a list of options with limit X,Y,Z or P (multiple regression). So Logistic regression is used in mathematical software to understand the relationship between dependent variables and one or more different variables by measuring probability using statistical regression calculations.

This form of study can assist you in assessing the risk of an occurrence or the selecting process. For example, you might want to know whether a visitor can select whether or not to accept an offer presented on your website (depending on variables). Visitors' well-known characteristics, such as the sites they frequent, redesign visits to ones site, and activity on your site, may be examined in your research (independent variables). The retrospective models assist you in determining the likelihood of which types of visitors will accept - or reject - the offer. As a results, you'll be able to make more informed decisions regarding your sponsorship or gift.

Mathematically logistic regression calculate multiple linear regression function:

$$\text{logit}(p)$$
$$= \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 \cdot x_{i1} + \beta_2 \cdot x_{i2} + \_ + \beta_p \cdot x_{in}$$

for i = 1…n .

Output =0 or 1

Hypothesis => Z = WX + B

hΘ(x) = sigmoid (Z)

Sigmoid Function

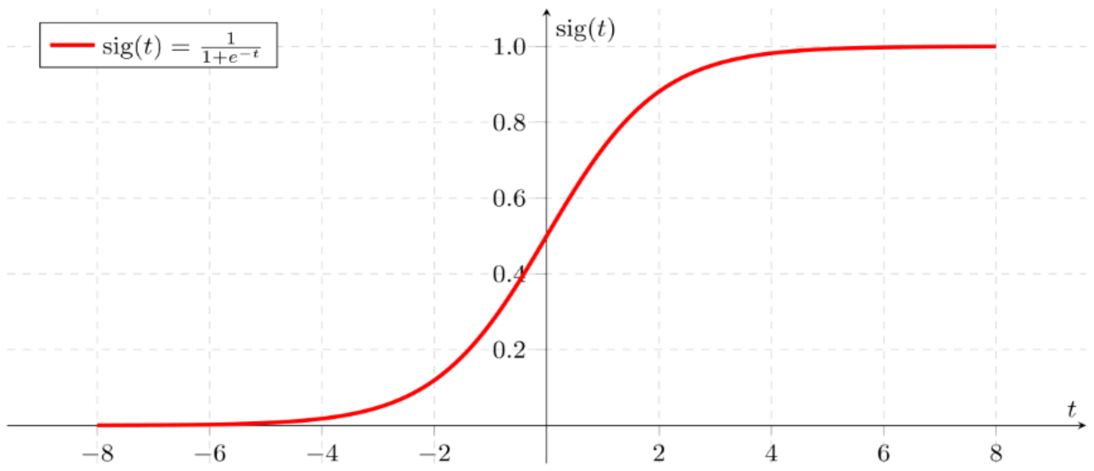


Figure 9: Sigmoid Function

Types of Logistic regression:
(i) Binary regression
(ii) Multinomial regression
(iii) Ordinal logistic regression

(i) Binary regression: It has only to outcomes like true or false or we can say 0 or 1.
For example: male or female.

(ii) Multinomial regression: In this type of regression there are three or more type of categories and there is no order of these categories.
For example: non-veg or veg or vegan.

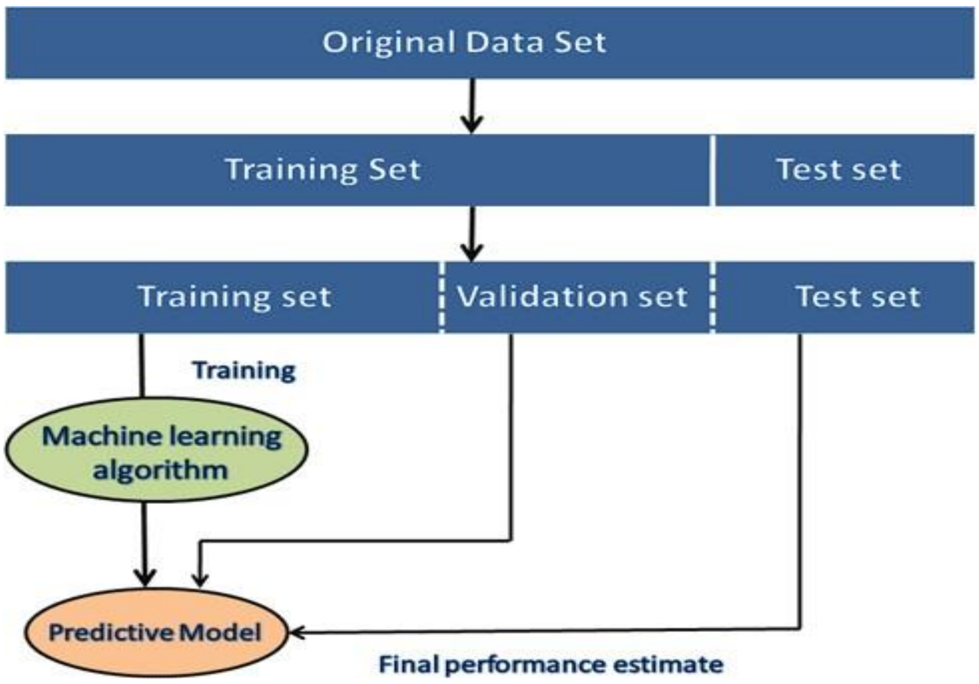
(iii) Ordinal Logistic regression: There is ordering in ordinal logistic regression. There are also three or more categories in ordinal logistic regression.
For example: Hotel rating from 1 to 7.

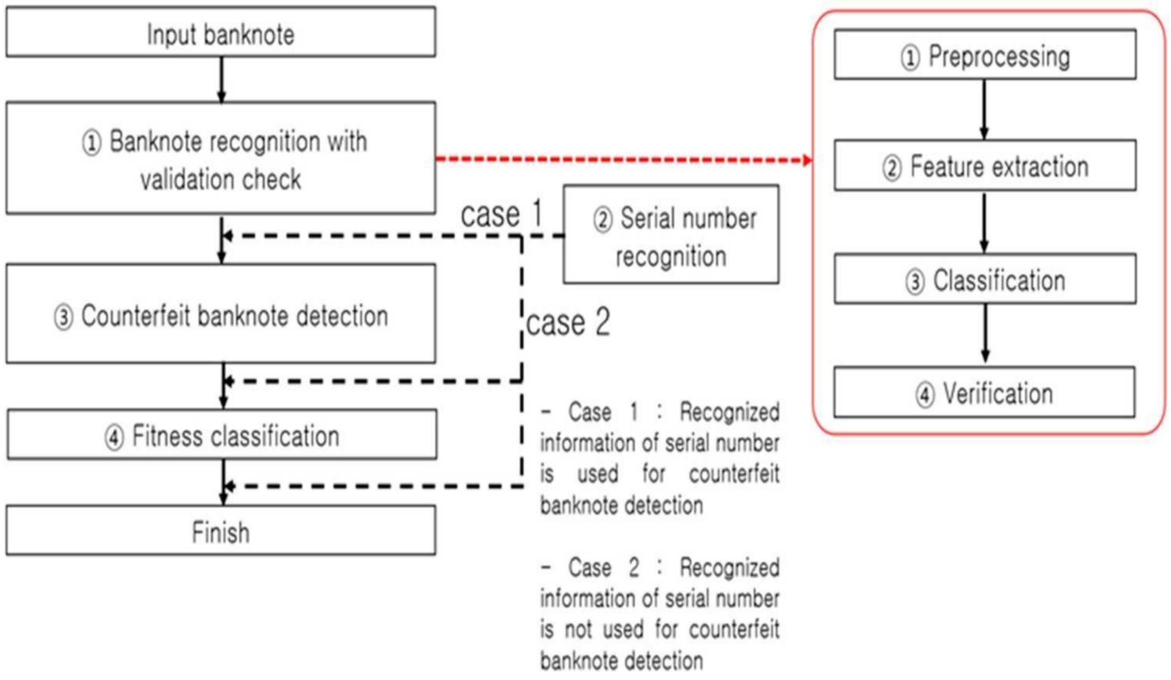
Over fitting: To check if the model is fit or not while selecting the model for the logistic regression analysis is also very important. Adding an independent variant to the retrospective model will always increase the value of the variance defined in log odd (usually expressed in R^2). However, additional flexibility in the model can lead to over-installation, which reduces model fulfillment beyond the data where the model is equal.

3.3 Model Development

State Transition Diagram of the Project

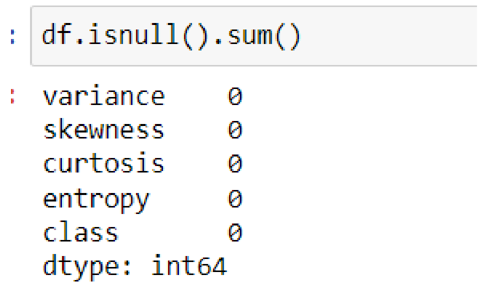
State of the data changes every time it passes through a model. Firstly, data is splited into two sets first is training set and second is test set using train test_split from sklearn module. Then in hyperparameter tuning data is divided into validation and training set. If the problem is of classification then by default stratified K fold cross validation is implemented which also take care of balancing the data. At last, final performance is evaluated using test set.




Input banknote
① Banknote recognition with validation check
case 1
② Serial number recognition
case 2
③ Counterfeit banknote detection
④ Fitness classification
Finish
- Case 1 : Recognized information of serial number is used for counterfeit banknote detection
- Case 2 : Recognized information of serial number is not used for counterfeit banknote detection
① Preprocessing
② Feature extraction
③ Classification
④ Verification
Figure 10: State Transition Diagrams

3.4 Analytical

In exploratory data analysis, we first checked whether our dataset contains any null value or not. Therewere no null values in our dataset.



```
: df.isnull().sum()

: variance    0
  skewness    0
  curtosis    0
  entropy     0
  class       0
  dtype: int64
```

Then we proceed to find out whether our dataset is normally distributed or not. We found out that therewas some skewness in our dataset.

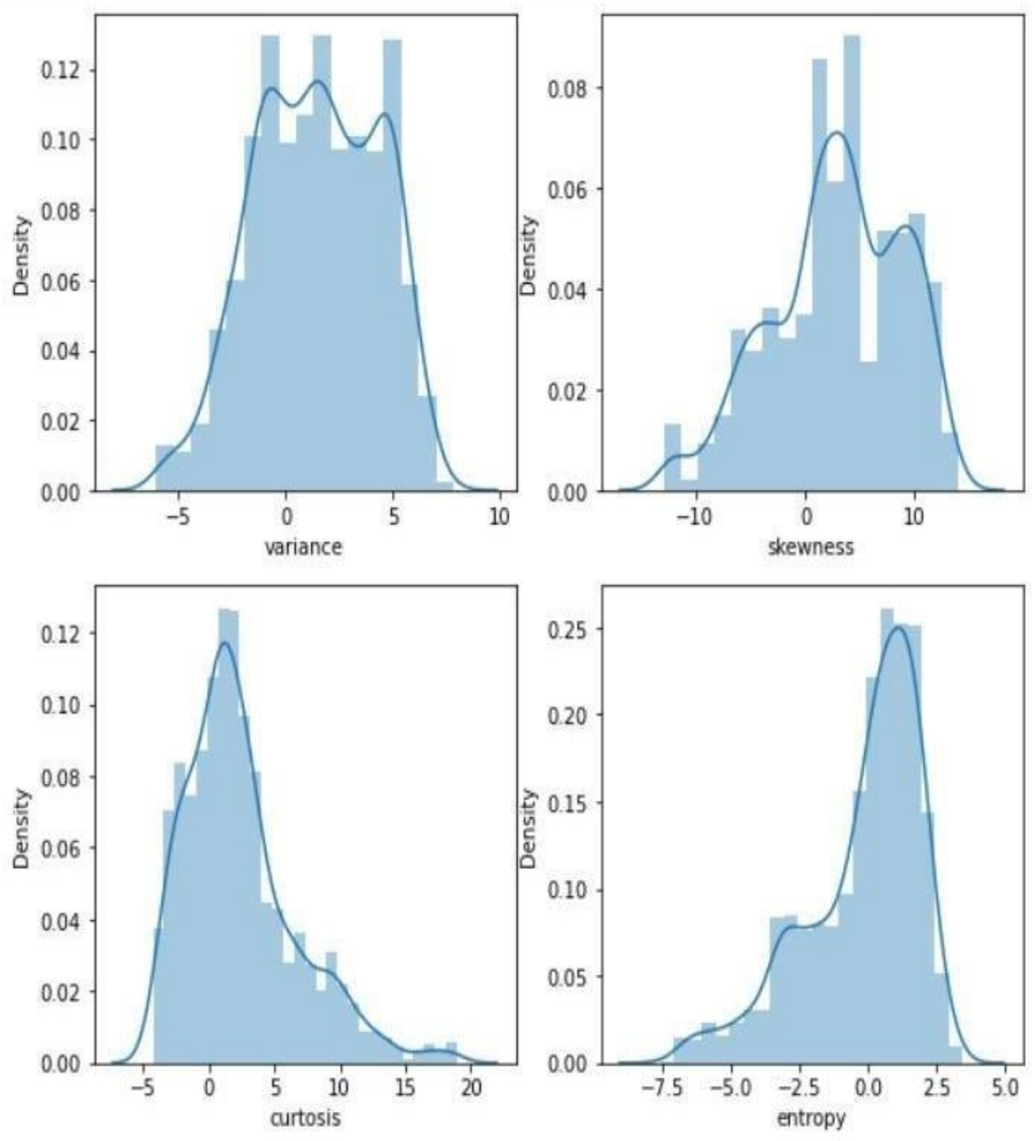


Figure 11: Normal Distribution of Features

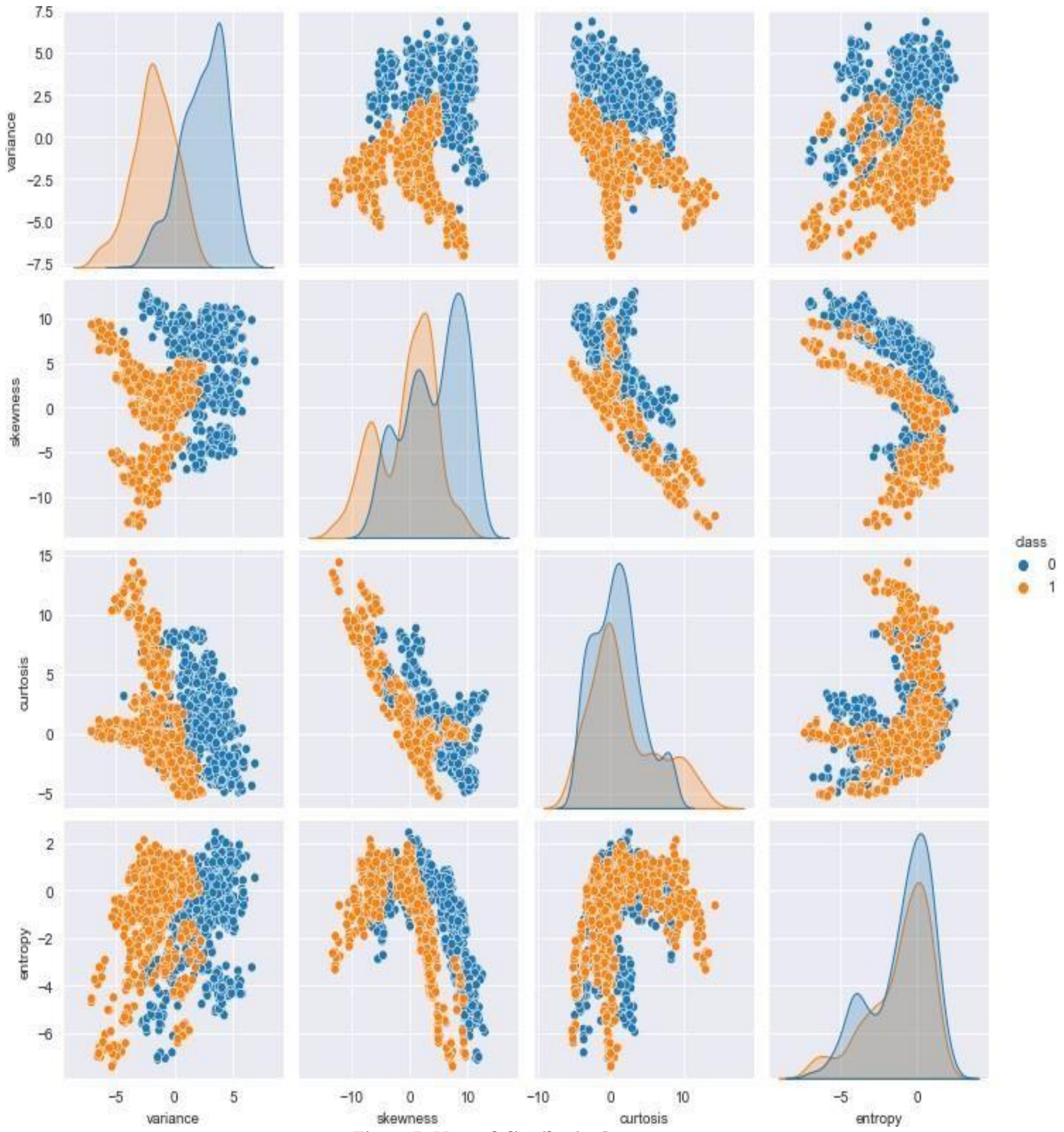


Figure 5: Normal distribution2

There are many ways to remove skewness from data. But we used log transformation to remove the skewness. It is basically taking log of each element in the dataset. But it may give error when taking log of 0. Below was the result we got:

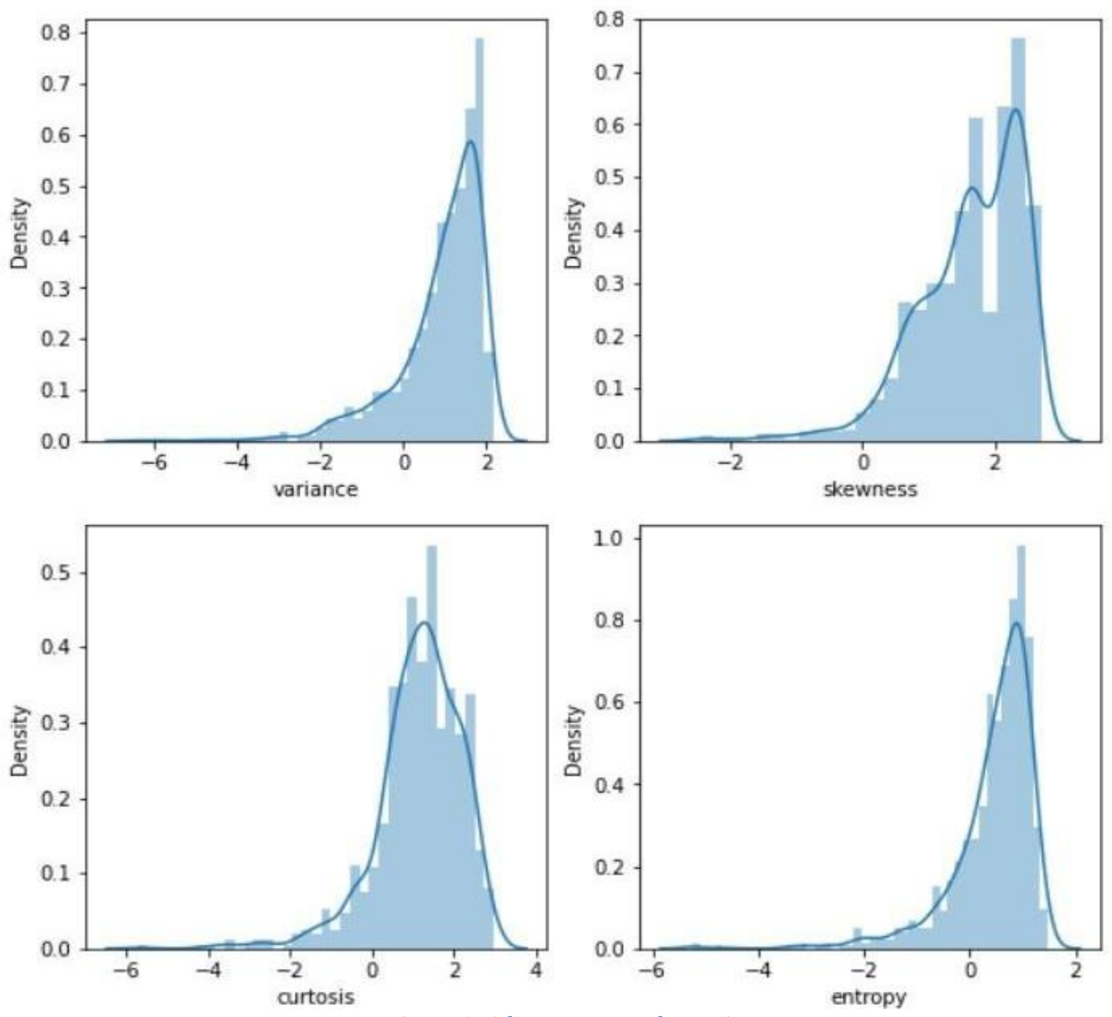
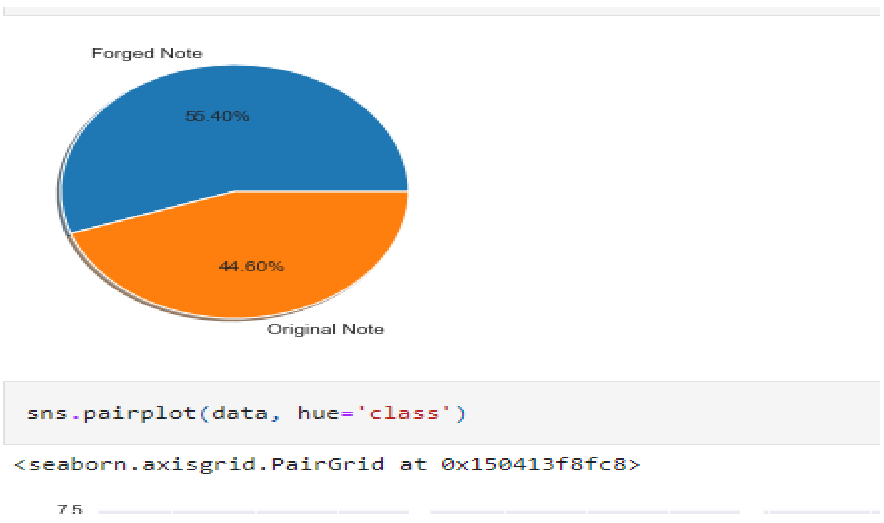


Figure 6: After Log Transformation

We can see that now our dataset is balanced.

Now, we will check whether our dataset is balanced or not in term of target classes.



```
sns.pairplot(data, hue='class')
```

```
<seaborn.axisgrid.PairGrid at 0x150413f8fc8>
```

Figure 7: Distributed or not -Target classes1

```
sns.countplot(df['class'])
```

```
C:\Users\tisha\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
  warnings.warn(
<AxesSubplot:xlabel='class', ylabel='count'>
```

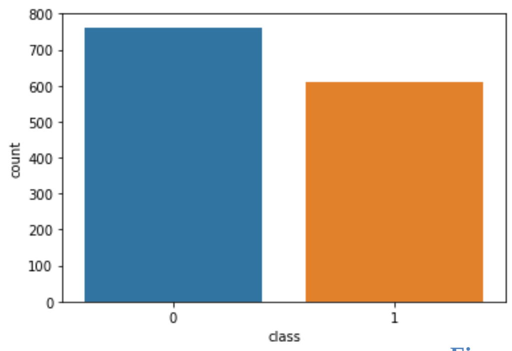
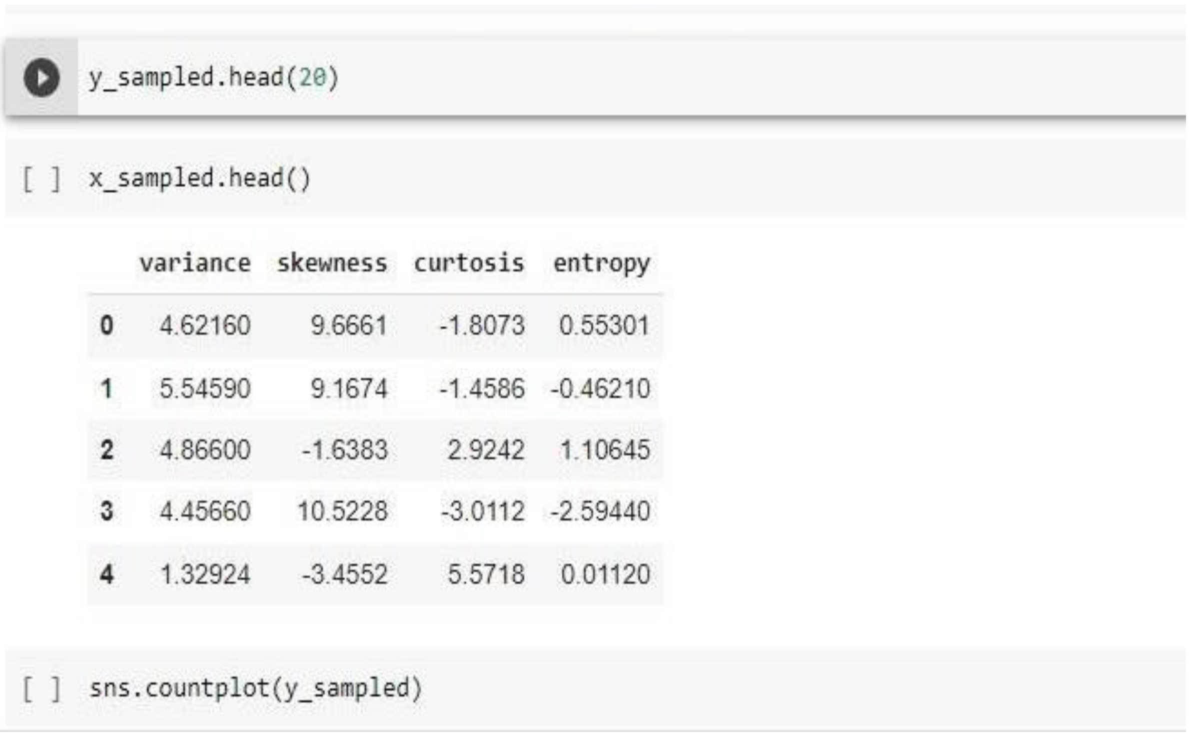


Figure 8 Dataset check

Dataset is almost balanced but to be safe we will apply Random Over Sampler to balance the dataset.There may be biasness in the data if data is imbalanced.

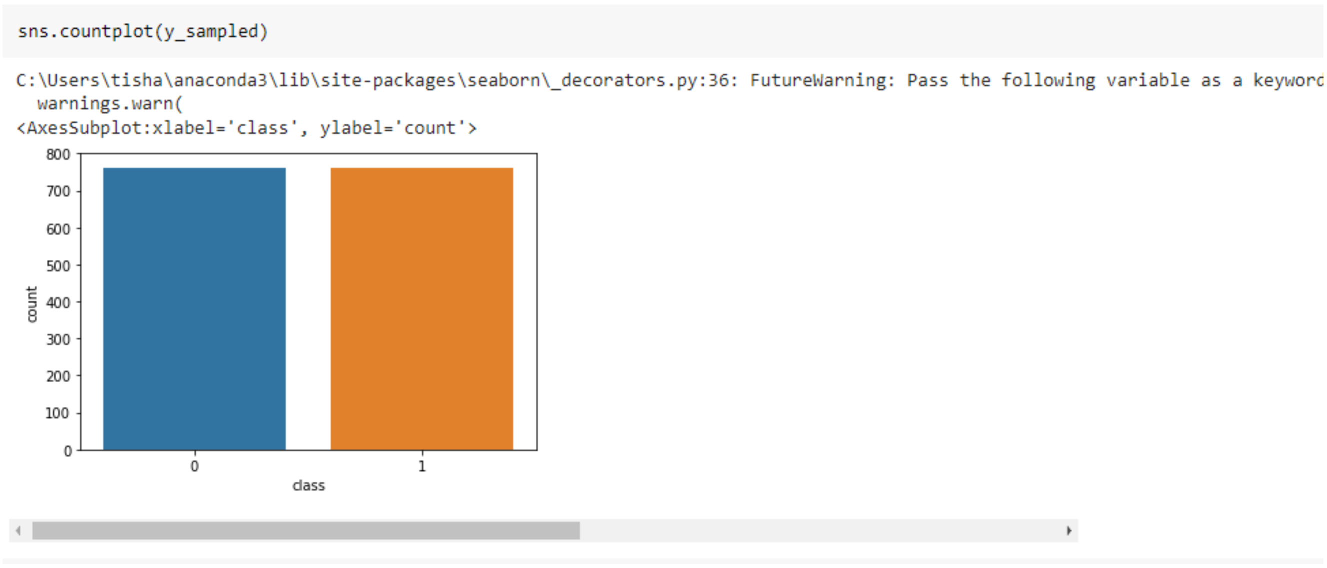


```
y_sampled.head(20)
```

```
[ ] x_sampled.head()
```

	variance	skewness	curtosis	entropy
0	4.62160	9.6661	-1.8073	0.55301
1	5.54590	9.1674	-1.4586	-0.46210
2	4.86600	-1.6383	2.9242	1.10645
3	4.45660	10.5228	-3.0112	-2.59440
4	1.32924	-3.4552	5.5718	0.01120

```
[ ] sns.countplot(y_sampled)
```



```
sns.countplot(y_sampled)
```

```
C:\Users\tisha\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyworc
  warnings.warn(
<AxesSubplot:xlabel='class', ylabel='count'>
```

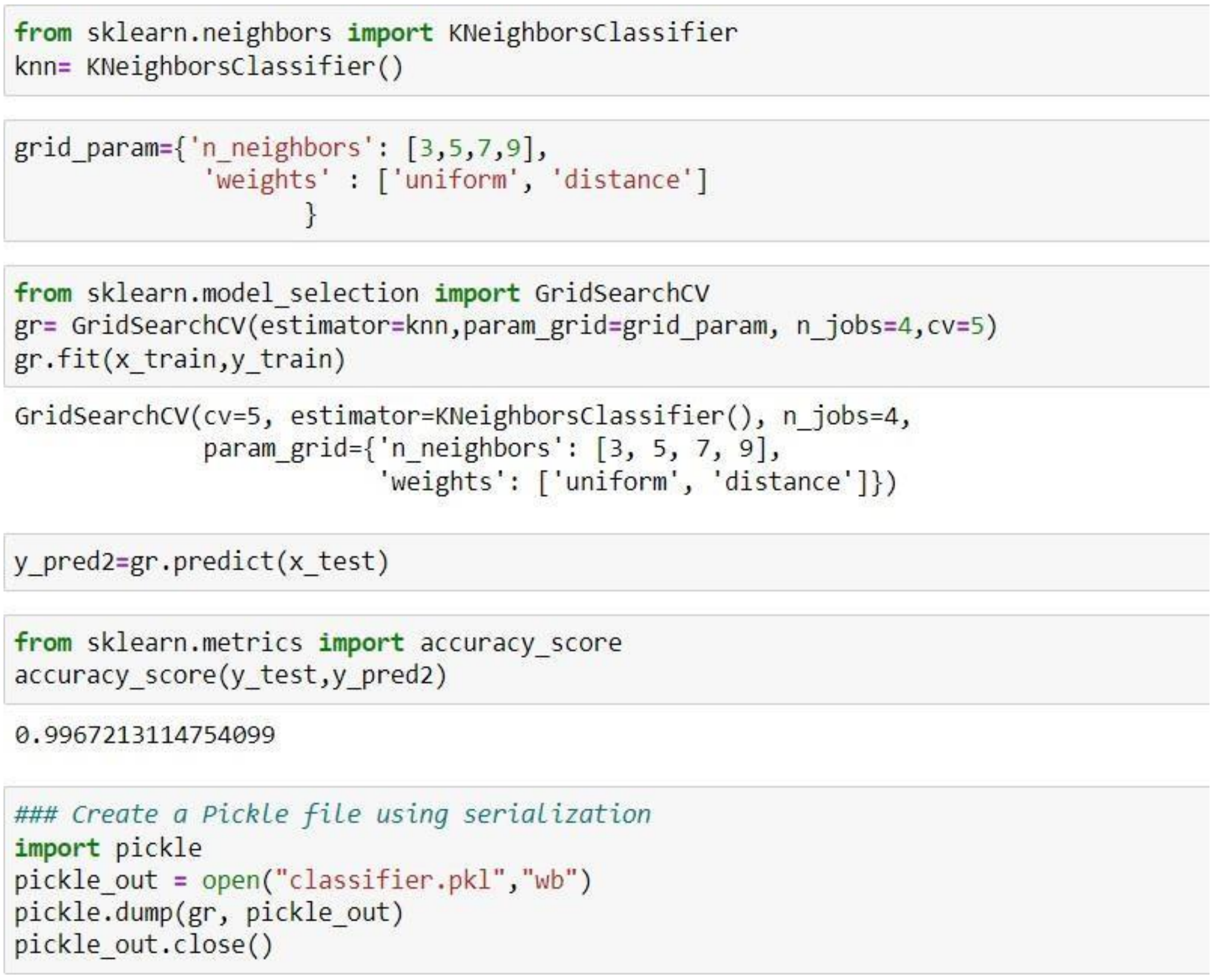
Figure 9: Balanced Dataset

Our dataset looks balanced now. We can create a model on top of this data.

3.5 Experimental

After performing exploratory data analysis we trained our model using KNN and Random forest and which algorithm gave better accuracy and stable model was given priority to be the base model. After that, we stored trained model in pickle file format. So, that we could use that in our web api.

Below is the code that we implemented:



```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier()
```

```
grid_param={'n_neighbors': [3,5,7,9],
             'weights' : ['uniform', 'distance']
                    }
```

```
from sklearn.model_selection import GridSearchCV
gr= GridSearchCV(estimator=knn,param_grid=grid_param, n_jobs=4,cv=5)
gr.fit(x_train,y_train)
```

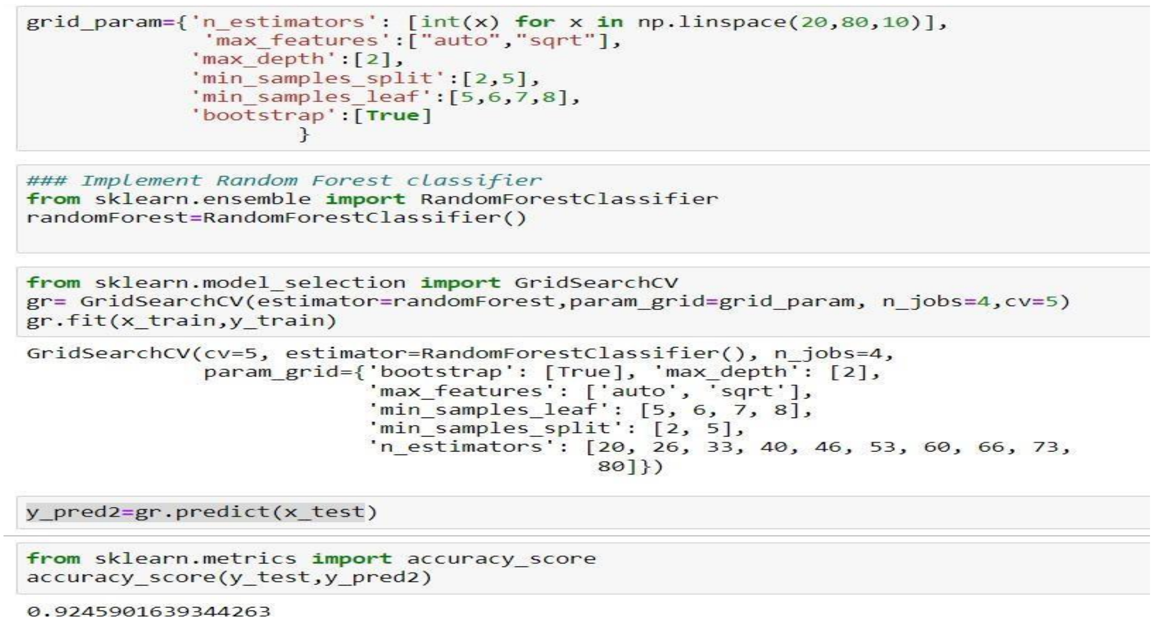
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(), n_jobs=4,
             param_grid={'n_neighbors': [3, 5, 7, 9],
                         'weights': ['uniform', 'distance']})
```

```
y_pred2=gr.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred2)
```

```
0.9967213114754099
```

```
### Create a Pickle file using serialization
import pickle
pickle_out = open("classifier.pkl","wb")
pickle.dump(gr, pickle_out)
pickle_out.close()
```



```
grid_param={'n_estimators': [int(x) for x in np.linspace(20,80,10)],
              'max_features':["auto","sqrt"],
             'max_depth':[2],
             'min_samples_split':[2,5],
             'min_samples_leaf':[5,6,7,8],
             'bootstrap':[True]
                    }
```

```
### Implement Random Forest classifier
from sklearn.ensemble import RandomForestClassifier
randomForest=RandomForestClassifier()
```

```
from sklearn.model_selection import GridSearchCV
gr= GridSearchCV(estimator=randomForest,param_grid=grid_param, n_jobs=4,cv=5)
gr.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=4,
             param_grid={'bootstrap': [True], 'max_depth': [2],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [5, 6, 7, 8],
                         'min_samples_split': [2, 5],
                         'n_estimators': [20, 26, 33, 40, 46, 53, 60, 66, 73,
                                          80]})
```

```
y_pred2=gr.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred2)
```

```
0.9245901639344263
```

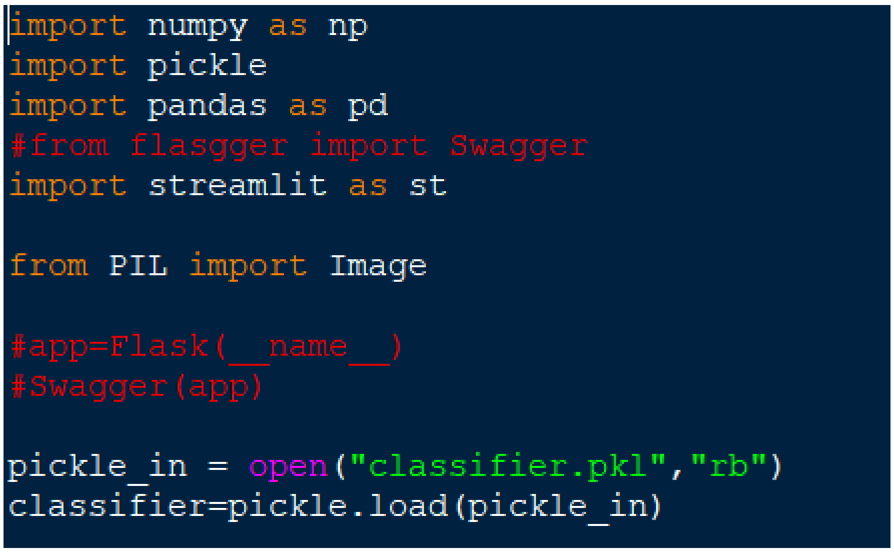
3.3.1 Flask api:
Flask is a very small and simple web app framework which is written in python. Many times developers used flask to build web applications.

Key points:-

- It is easy to use.
- It is to fix and find the mistakes and errors.

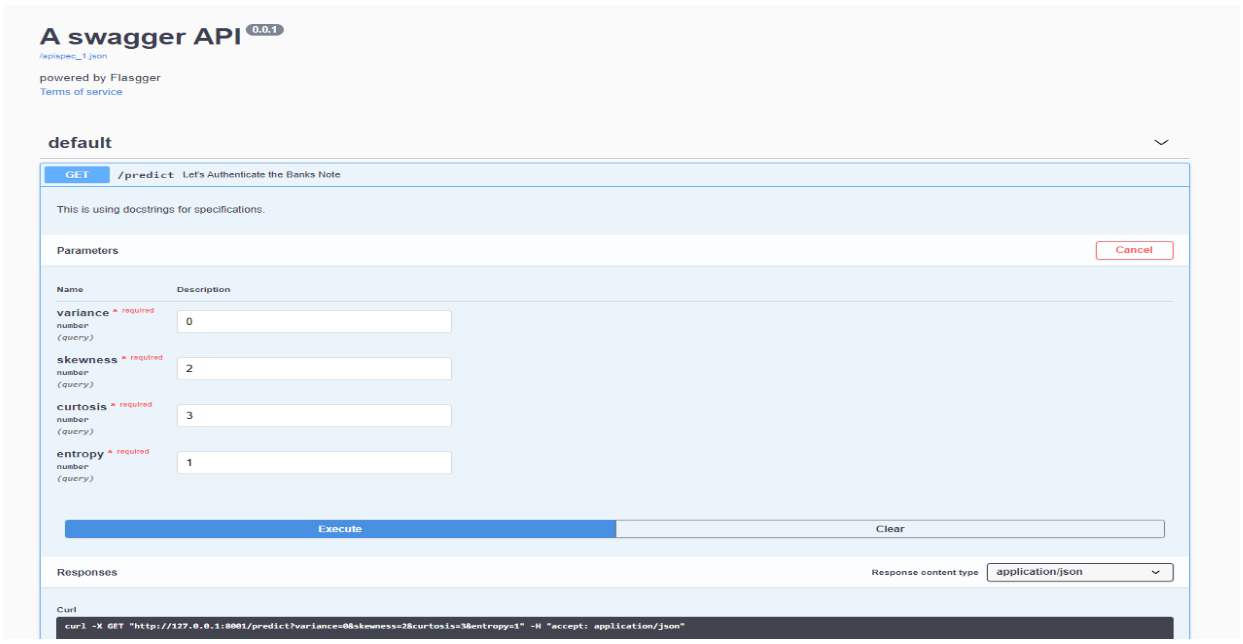
Flask features:
- It has its own development server to run any applicaton. Server is only for development not for deployment.
- Flask supports Jinja2 .
- It is based on Werkzeug WSGI tool .
- Flask provides extensive documentation.

Importing flask



```
import numpy as np
import pickle
import pandas as pd
#from flasgger import Swagger
import streamlit as st

from PIL import Image

#app=Flask(__name__)
#Swagger(app)

pickle_in = open("classifier.pkl","rb")
classifier=pickle.load(pickle_in)
```

A swagger API: It allows us to describe the structure of our application user interface simply api so that it is easily readable by machines.


A swagger API 0.0.1

/apispec_1.json

powered by Flasgger

Terms of service

default

GET /predict Let's Authenticate the Banks Note

This is using docstrings for specifications.

Parameters

Name	Description
variance * required number (query)	0
skewness * required number (query)	2
curtosis * required number (query)	3
entropy * required number (query)	1

Responses

Response content type: application/json

Curl

```
curl -X GET "http://127.0.0.1:8001/predict?variance=0&skewness=2&curtosis=3&entropy=1" -H "accept: application/json"
```

Figure 10: A Swagger API

Features calculation: Here I have calculated the features like skewness, kurtosis, entropy and variance of the bank note image using the inbuilt functions. Below is the snapshot of the code:

(i) Skewness: It tells us about the symmetry. It is the measure of or we can say lack of symmetry. The histogram is negatively skewed if the skewness is negative that means its tail is left tail longer than its right one.

Below is the graph that show us about the skewness:

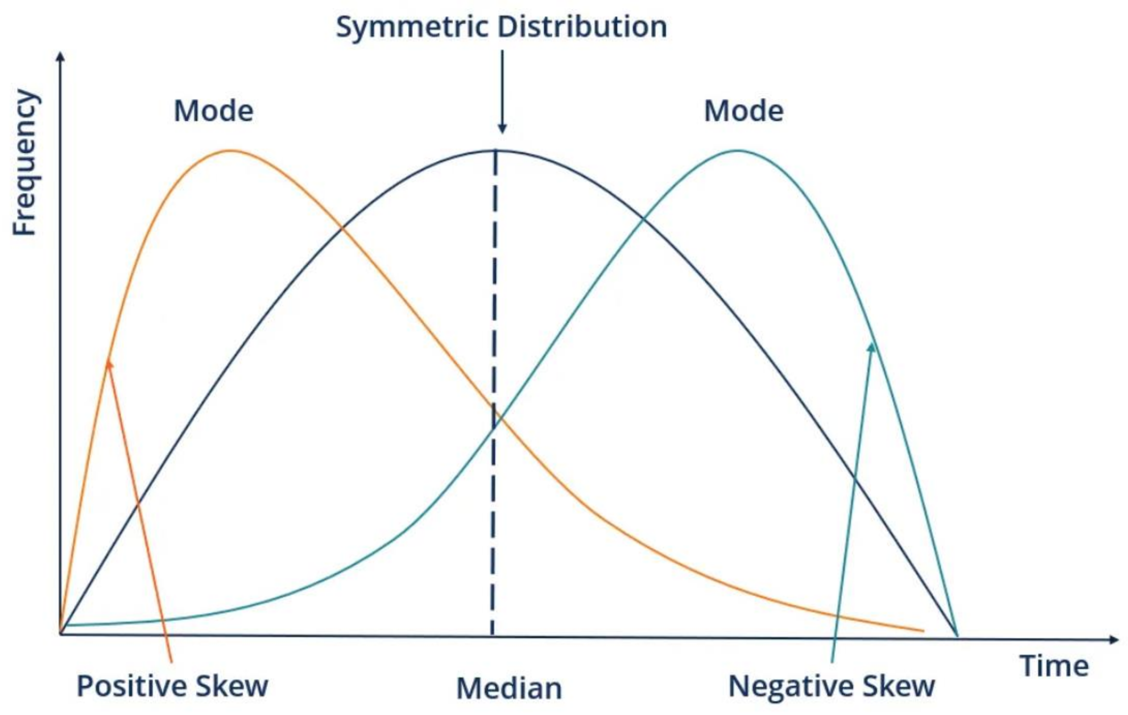
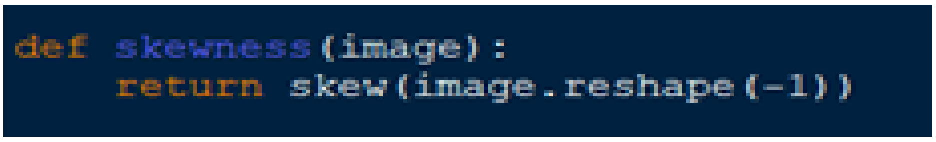


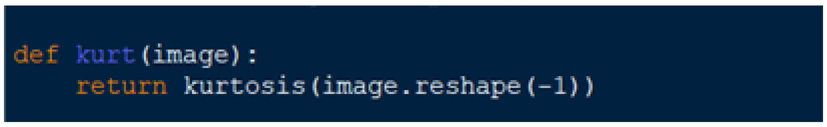
Figure 11: Skewness Graph



```
def skewness(image):
    return skew(image.reshape(-1))
```

(ii) Kurtosis: Kurtosis measure whether the data are flat with respect to the normal distribution .This is data sets with high kurtosis tend to have a distinct peak near the mean decline rather rapidly and heavy tails.

Data sets with low kurtosis tend to have a flat top near the mean rather than a sharp peak.



```
def kurt(image):
    return kurtosis(image.reshape(-1))
```

Kurtosis :

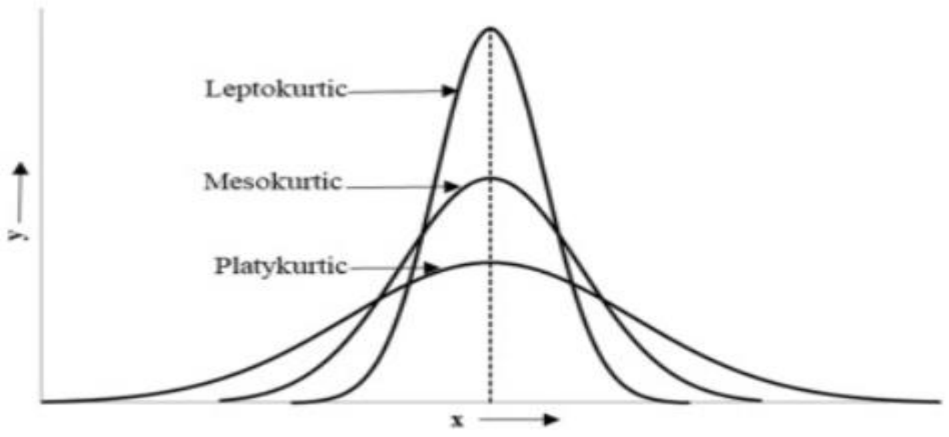
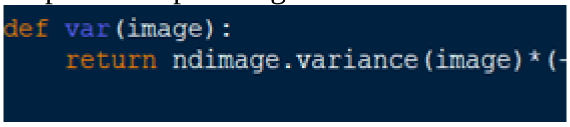


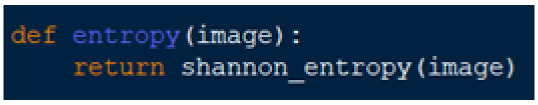
Figure 12:Kurtosis

(iii) Variance: It represents the squares to the standard deviations that correspond to the output and input images values.



```
def var(image):
    return ndimage.variance(image)*(
```

(iv) Entropy: It is the randomness of the information present in the image. It tells us how the pixels are present in the image .The entropy of image can be calculated at each pixel position like i, j the entropy of the pixel values within a two dimension region centered at i, j .



```
def entropy(image):
    return shannon_entropy(image)
```

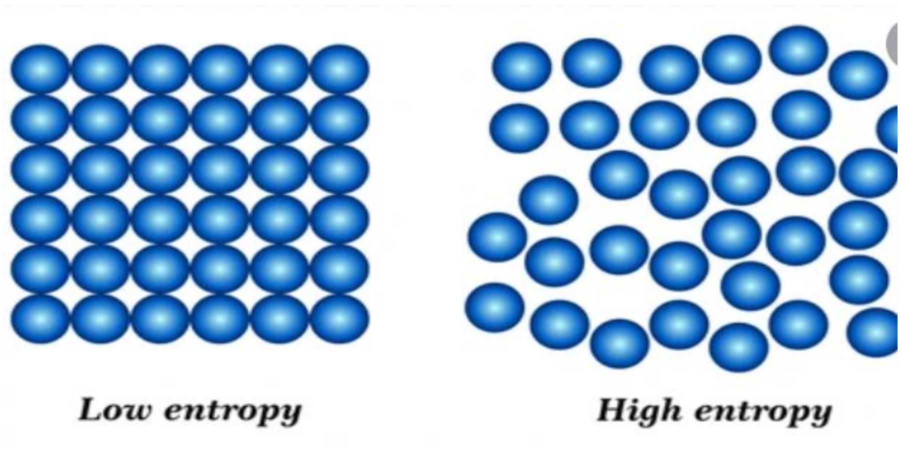


Figure 13: Entropy

Calculation of skewness, entropy, variance and kurtosis:

```
shannon_entropy(image)
skew(image.reshape(-1))

kurtosis(image.reshape(-1))

ndimage.variance(image)*(-10)
```

Streamlit:

Streamlit is an open source source for Python library. It is very easy to use and with the help of streamlit we can create attractive and beautiful web applications for any machine learning and data science project. It is very easy to build UI very quickly with the help of the powerful and simple Streamlit app model.

There are several widgets available in streamlit, such as st.selectbox, st.checkbox, st.slider, and submit box. Let's see an example of a widget in the light. For the first time, the web application will output a “0 square by 0” character, and then when the user raises or lowers the widget, the code is restarted by streaming from top to bottom and providing the current status of the widget variable.

Initially , the web app ask for the input image and after loading or dropping the image in the app than the image is processed to extract the required features which are skewness, kurtosis, entropy and variance from the input image. These features are extracted using the inbuilt functions from various python libraries like numpy, pillow and open-cv.
The extracted features are than displayed in the app using the st.text_input(). Normally st.text_input() is used to take input but the reason I am using this instead of st.text() which is used to display text is st.text_input() provides us the functionality or options to edit the extracted features so that we can check on custom inputs or instead of using image input we can use this to directly enter the value of features.
These features than are passed to the model to make prediction. I have used Random forest

in my model to make the predictions. More details or better explanation about the model and the process are discussed in this report above this section.

Streamlit uses a python script from top to bottom. Each time a user shares a script it is restarted from top to bottom. Streamlit allows you to use a repository of expensive tasks such as uploading large data sets.

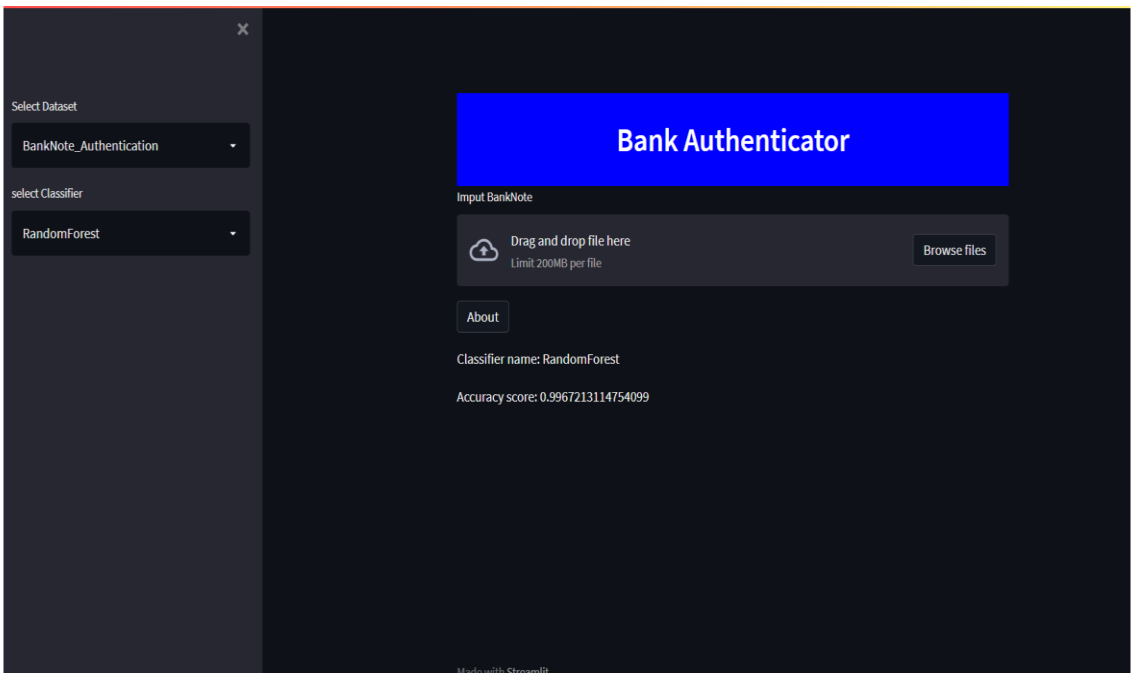
Now execute the following command in cmd:

Use **streamlit run app.py**

Key Features:

- In a few minutes, Streamlit converts data documents into shared web applications.
- Everything is written in Python. Everything is free.
- No need for prior experience.
- To easily share, care for, and participate in your apps.

Front end of web api using streamlit.

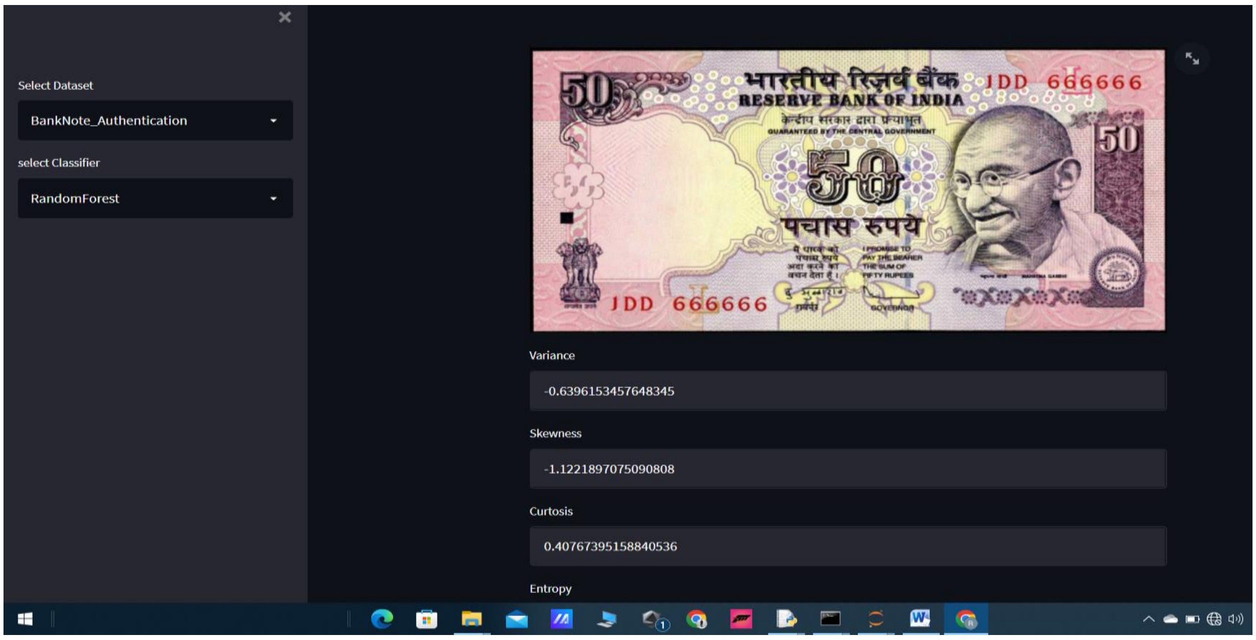


Classifier name: RandomForest

Accuracy score: 0.9967213114754099

Figure 14: Streamlit-Front end of Web api

After dropping file or Image:


Select Dataset

BankNote_Authentication

select Classifier

RandomForest

Variance

-0.6396153457648345

Skewness

-1.1221897075090808

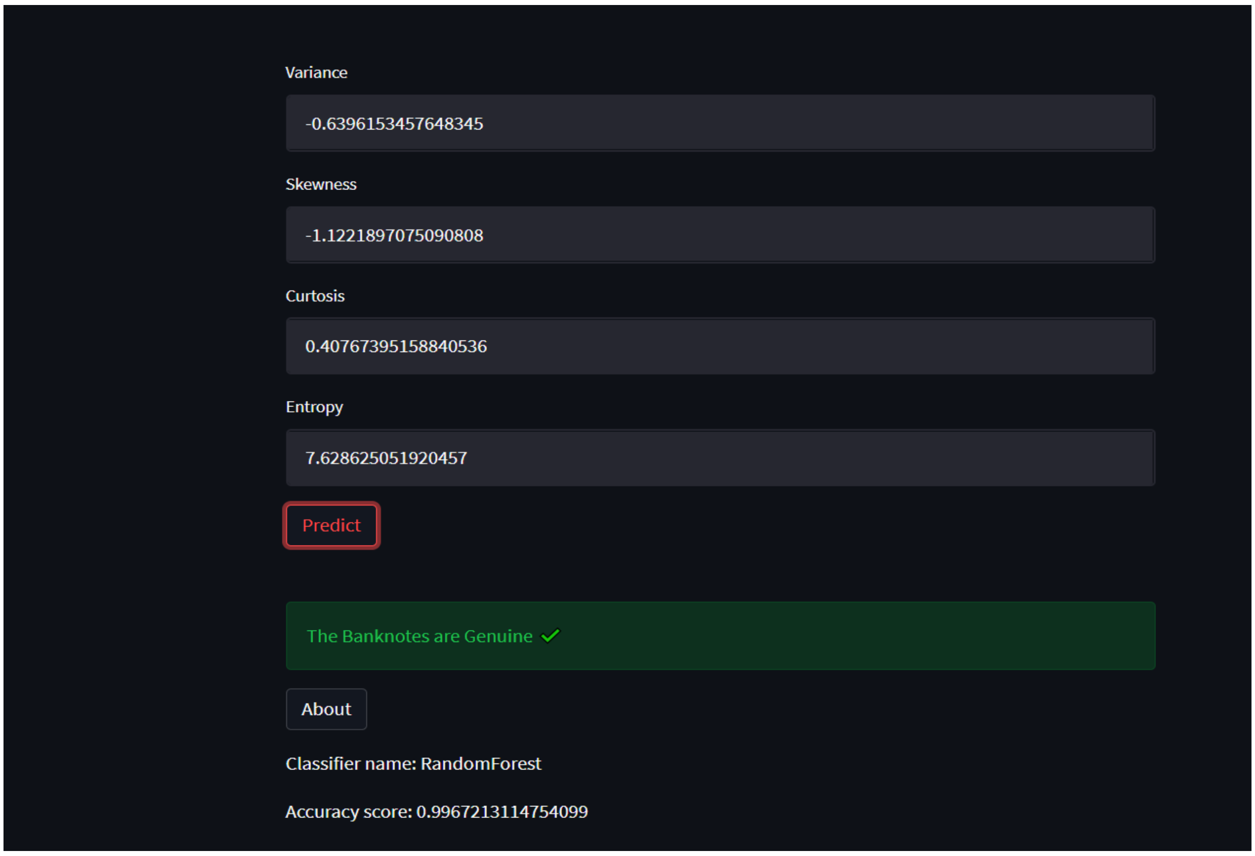
Curtosis

0.40767395158840536

Entropy

Figure 15: Screenshot of webapp after dragging image

After clicking on predict button:


Variance

-0.6396153457648345

Skewness

-1.1221897075090808

Curtosis

0.40767395158840536

Entropy

7.628625051920457

The Banknotes are Genuine ✓

Classifier name: RandomForest

Accuracy score: 0.9967213114754099

Figure 16: Screenshot of Final output

3.6 Mathematical

Mathematics Involved in XGBoost: XGBoost does not use entropy or Gini indices. Instead, it utilizes gradient (the error term) and hessian for creating the trees. Hessian for a Regression problem isthe *number of residuals* and for a classification problem. Hessian is the 2nd order derivative of the loss.

$$h_m(x) = \frac{\partial^2 L(Y, f(x))}{\partial f(x)^2}_{f(x)=f^{(m-1)}(x)}.$$

here **L** =loss function.

Initialize the tree with only one leaf.
compute the similarity using the formul

$Similarity{=}Gradient^2/hessian{+}\lambda$

Where λ is the regularization term.

Now for splitting data into a tree form, calculate

$Gain{=}leftsimilarity{+}rightsimilarity{-}similarityforroot$

For tree pruning, the parameter γ is used. The algorithm starts from the lowest level of the treeand then starts pruning based on the value of γ.

If $Gain{-}\gamma{<}0$, remove that branch. Else, keep the branch

Learning is done using the equation

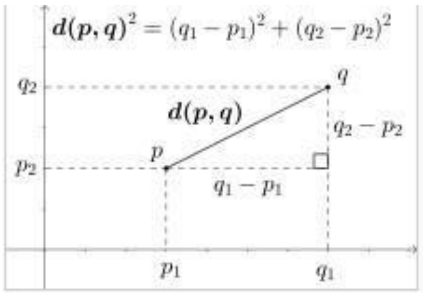
$NewValue{=}oldValue{+}\eta{*}prediction$

where η is the learning rate

Mathematics involved in K Nearest Neighbourhood:

Euclidean Distance:
It is the most commonly used method to calculate the distance between two points. The

Euclideandistance' is calculated as:



$$d(\mathbf{p},\mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

Similarly, for n-dimensional space, the Euclidean distance is given as:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Hamming distance

Used to calculate the number of mismatching points between two vectors.

$$HamD(x,y) = \sum_{i=1}^{n} 1_{x_i \neq y_i}$$

Generally, if we have features as categorical data then we consider the difference to be 0 if both thevalues are the same and the difference is 1 if both the values are different.

Manhattan Distance

Used to calculate the mode of differences between the different values in vectors.

$$MD(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

This is less influenced by outliers than the Euclidean distance. With very high dimensional data it ismore preferred.

Chapter 4: PERFORMANCE ANALYSIS

4.1 Accuracy Formula

The following measures were used to measure the performance of the implemented models: Accuracy - The accuracy of the test is the ability to accurately distinguish between genuine andcounterfeit note test cases.

Accuracy = (tp+tn) / (tp+tn+fp+fn)

Sensitivity - Sensitivity of the test is the ability to accurately identify the original note cases.
Sensitivity = tp / (tp+fn)

Specificity - The uniqueness of the test is the ability to accurately identify cases of counterfeitnotes.

Specifcity = tn / (tn+fp)

Accuracy - The accuracy of the test is the ability of the classifier to determine whether the noteslabeled as genuine are genuine.

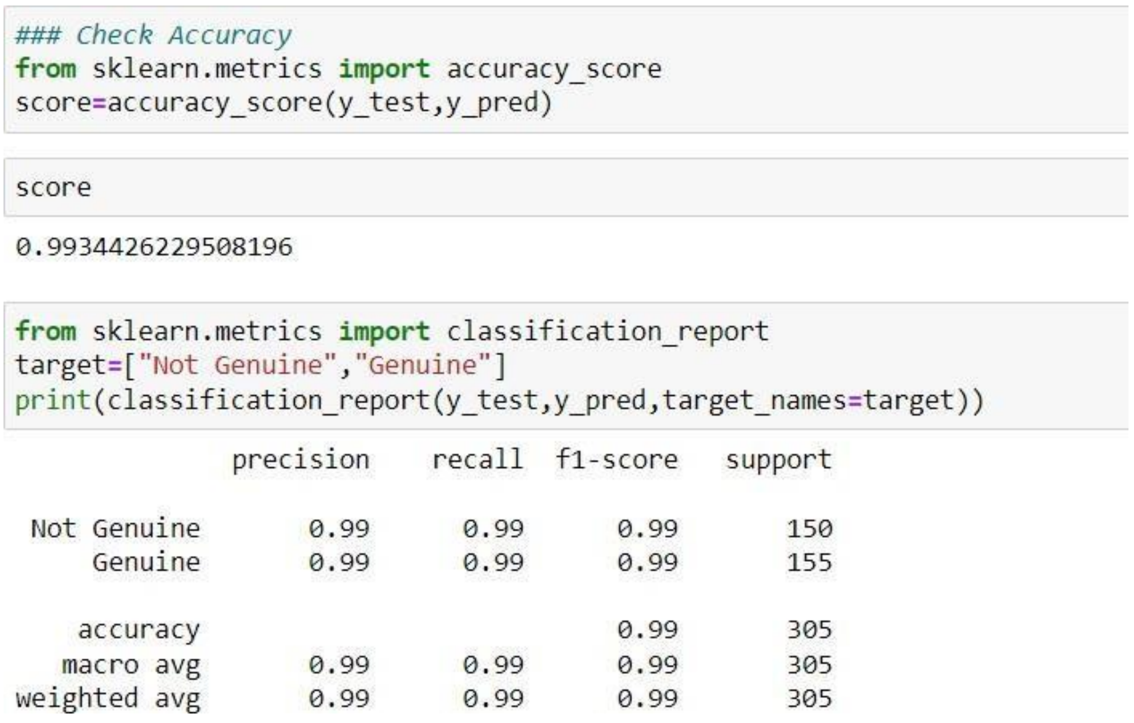
Accuracy = tp / (tp+fp)Where,

True positive = Number of cases correctly determined as original notes. True negative = Number of cases correctly determined as counterfeit notes.False positive = Number of cases misidentified as original notes.

False negative = Number of cases misidentified as counterfeit notes.
We will compare two algorithms Random forest classifier and K Nearest Neighborhood.

4.2 Analysis of Random Forest

Random forest is the bagging algorithm which uses multiple decision trees and most frequent prediction will be the output. Below are the results we got with random forest without hyper parameter tuning.



```
### Check Accuracy
from sklearn.metrics import accuracy_score
score=accuracy_score(y_test,y_pred)
```

```
score
```

```
0.9934426229508196
```

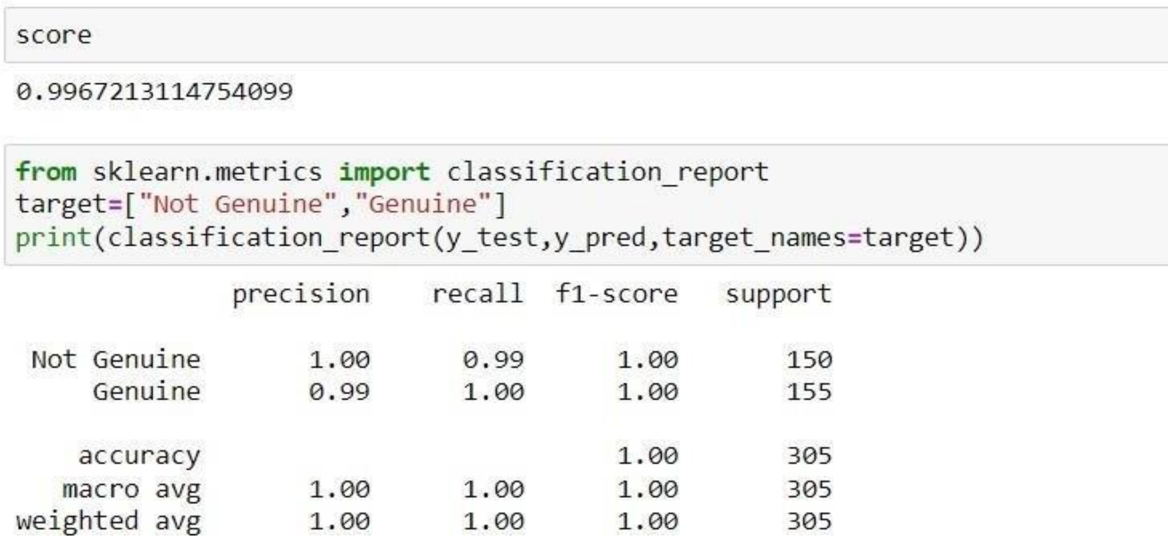
```
from sklearn.metrics import classification_report
target=["Not Genuine","Genuine"]
print(classification_report(y_test,y_pred,target_names=target))
```

```
              precision    recall  f1-score   support

 Not Genuine       0.99      0.99      0.99       150
     Genuine       0.99      0.99      0.99       155

    accuracy                           0.99       305
   macro avg       0.99      0.99      0.99       305
weighted avg       0.99      0.99      0.99       305
```

4.3 Analysis of K Nearest Neighborhood

Given a dataset with different classes, this algorithm predicts by calculating the Euclidean orManhattan distance between the testing set and every training set.

Below are the results of K Nearest



```
score
```

```
0.9967213114754099
```

```
from sklearn.metrics import classification_report
target=["Not Genuine","Genuine"]
print(classification_report(y_test,y_pred,target_names=target))
```

```
              precision    recall  f1-score   support

 Not Genuine       1.00      0.99      1.00       150
     Genuine       0.99      1.00      1.00       155

    accuracy                           1.00       305
   macro avg       1.00      1.00      1.00       305
weighted avg       1.00      1.00      1.00       305
```

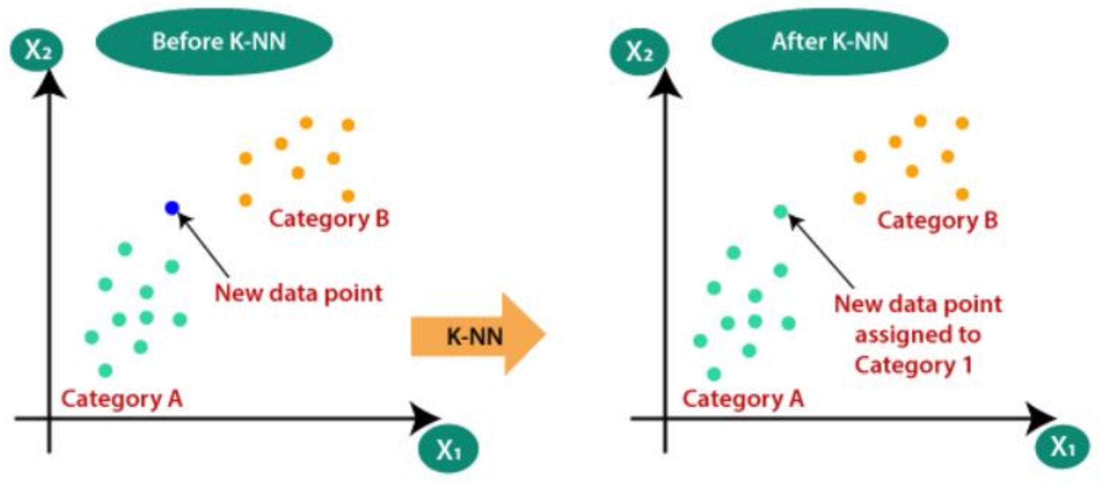
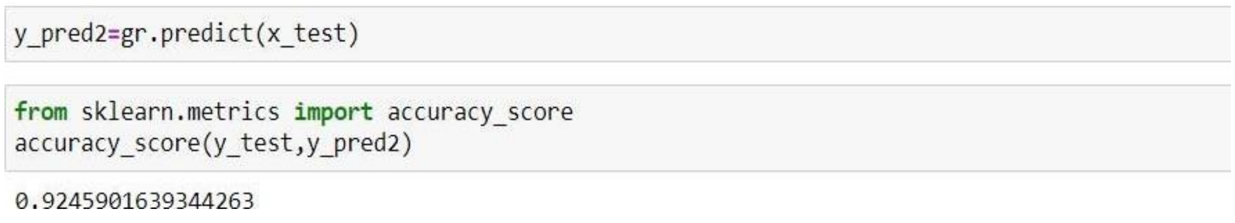


Figure 17: KNN

Both algorithms gives very good accuracy. But accuracy is too high. There may be a overfitting problem. Overfitting problem occurs when algorithm gives very good accuracy with training data but bad accuracy with testing data i.e. high variance and low bias.
So, we also checked accuracy of algorithms after hyperparameter tuning. Below are the results wegot for Random forest.

4.4 Accuracy



```
y_pred2=gr.predict(x_test)
```

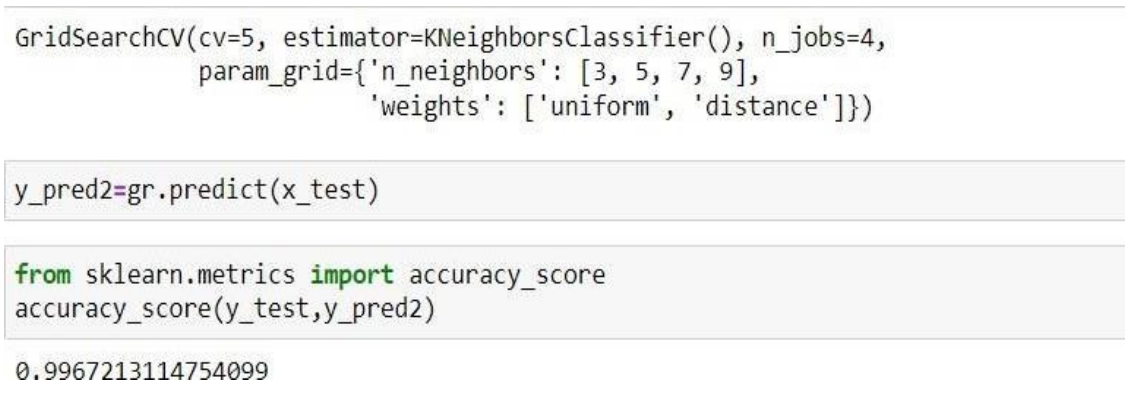
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred2)
```

```
0.9245901639344263
```

4.5 Accuracy of K Nearest Neighborhood after Hyper Parameter Tuning

From above results we can say that K Nearest Neighborhoodis giving same accuracy after and before hyper parameter tuning. So I am not using K Nearest Neighborhood to make the prediction. Even though the accuracy of KNN is high we have to use Random forest classifier.

But in case of Random forest after hyper parameter tuning there is an decrease in accuracy score. So, we can say that even accuracy is high in case of KNN but Random forest is giving over all stable model. So, we have used Random forest in our web api as a base model.



```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(), n_jobs=4,
             param_grid={'n_neighbors': [3, 5, 7, 9],
                         'weights': ['uniform', 'distance']})
```

```
y_pred2=gr.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred2)
```

```
0.9967213114754099
```

Chapter 5 CONCLUSION

5.1 Conclusions

After analyzing the various methods used to detect counterfeit notes, the OR report provides banknote authentication using two supervised practice methods to identify banknotes as genuine or counterfeit. Extensive experiments have been conducted on banknote datasets using two models to calculate the best model suitable for the classification of notes. Accuracy is measured by accuracy, recall, f1 score values and we get almost the same accuracy. The result shows that Random forest outperforms K Nearest Neighborhood after hyper parameter tuning. Hence, we are successfully able to build a classification methodology to predict whether Bank note is genuine or not.

5.2 Future Scope

We would try out new machine learning algorithms and try to improve factors like accuracy.We will complete its web development part. We will use javascript, HTML, CSS at front end to create UI. We will also deploy this project on Heroku, aws or various cloud platforms. So that it will be publicly available.

This application is of great importance us the fraudulent groups will come out with ways to create fake banknotes that will fool many detection algorithm. We could try to use IR cameras that will be able to detect minute features like notes threading, holographs, color changing patterns and more. Although finding such dataset will be really tough and possibly only banks might have access to such dataset, nonetheless our algorithm can be trained on such a dataset to further improve the accuracy of the system. More over object detection algorithms like YOLO can be customized to detect low-level features using IR cameras. Such algorithm can be trained to find general features that genuine notes have and we can add a threshold to decide if the note is genuine or not.

5.3 Applications

There are various applications of machine learning algorithms but talking specifically about this project given the values of Kurtosis, variance, entropy, skewness one can easily identify whether the given note is genuine or not.

After some improvement this type of project can be very useful in banking sector, ATMs. Forged notes can be easily identified.

Overall this project may help not only in the banking sector but growth of overall country aswe know that bank notes are one of the most important aspects of a country

REFERENCES

1. AurélienGéron (2017), Hands–On ML with Scikit–Learn and TensorFlow (2nd Edition),Publisher: O'Reilly Media

2. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013), An Introduction to Statistical Learning: with Applications in R (1st edition),Publisher: Springer

3. Polat, Kemal, Seral Şahan, and SalihGüneş. "A novel hybrid method based on artificial immune recognition system (AIRS) with fuzzy weighted pre-processing for thyroid disease diagnosis." Expert Systems with Applications, Vol 32, Issue 4,May 2007.

4. Dogantekin, E., Dogantekin, A., and Avci, D., "An expert system based on generalized discriminant analysis and wavelet support vector machine for diagnosis of thyroid diseases." Expert Syst. Appl. , Vol. 38, Issue 1, 2011.

5. Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel (2010), Sci-kit learn,20-03-2021 from 'Scikit-learn: Machine Learning in Python', https://scikit-learn.org/stable/ .

6. Prachi Damodhar Shahare and Ram Nivas Giri, "Comparative Analysis of Artificial Neural Network and Support Vector Machine Classification for Breast Cancer Detection", International Research Journal of Engineering and Technology, Dec-2015.

7. Fumiaki Takeda, Lalita Sakoobunthu and Hironobu Satou, "Thai Banknote Recognition Using Neural Network and Continues Learning by DSP Unit", International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, 2003.

8. Costas Nastoulis, Apostolos Leros, and Nikolaos Bardis, "Banknote Recognition Based On Probabilistic Neural Network Models", Proceedings of the 10th WSEAS International Conference on SYSTEMS, Vouliagmeni, Athens, Greece, July 10-12, 2006.

9. Sigeru Omatu, Michifumi Yoshioka and Yoshihisa Kosaka, "Bank Note Classification Using Neural Networks", IEEE, 2007.

10. Swati V. Walke and Prof. Dr. D. M. Chandwadkar, “Counterfeit Currency Recognition Using SVM With Note to Coin Exchanger”, International Journal of Modern Trends in Engineering and Research, July 2015.

11. Sharmishta Desai, Shraddha Kabade, Apurva Bakshi, Apeksha Gunjal, Meghana Yeole, “Implementation of Multiple Kernel Support Vector Machine for Automatic Recognition and Classification of Counterfeit Notes”, International Journal of Scientific & Engineering Research, October-2014.

12. Masato Aoba, Tetsuo Kikuchi, and Yoshiyasu Takefuji, “Euro Banknote Recognition System Using a Threelayered Perceptron and RBF Networks”, IPSJ Transactions on Mathematical Modeling and it's BPN SVM 80 85 90 95 100 105 110 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 Sensitivity 100-Specificity ROC Curve for Hold-out Method International Journal of Computer Applications (0975 – 8887) Volume 179 – No.20, February 2018 26 Applications, May 2003.