Bank Note Authentication Web APP using

Python, Machine Learning, Github and Streamlit

### AN INTERNSHIP REPORT

*Submitted by*

**SURAJ RAVISHANKAR YADAV**

**[EC2331201010110]**

**Under the Guidance of**

## Prof. Muthuselvam

(Assistant Professor, Directorate of Online Education)

*in partial fulfillment for the award of the degree of*

**BACHELOR OF COMPUTER APPLICATIONS**

**(Specialization in Data Science)**



DIRECTORATE OF ONLINE EDUCATION

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

JUNE 2024

DIRECTORATE OF ONLINE EDUCATION

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

**BONAFIDE CERTIFICATE**

This internship report titled **“Bank Note Authentication Web APP using Python, Machine Learning, Github and Streamlit”** is the Bonafide work of **“SURAJ RAVISHANKAR YADAV [EC2331201010110]”**, who carried out the internship work under my supervision along with the company mentor. Certified further, that to the best of my knowledge the work reported herein does not form any other internship report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

# 

# INTERNSHIP OFFER LETTER

# 

# 

# INTERNSHIP COMPLETION CERTIFICATE

# 

# COURSE COMPLETION CERTIFICATE



**ACKNOWLEDGEMENTS**

I express my humble gratitude to **Dr. C. Muthamizhchelvan**,

Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. I extend my sincere thanks to Director DOE, SRM Institute of Science and Technology, **Dr. Manoranjan Pon. Ram,** for his invaluable support. I want to convey my thanks to Programme Coordinator **Prof. Muthuselvam**, Directorate of online Education, SRM Institute of Science and Technology, for his inputs during the project reviews and support throughout the project work. Once again, my inexpressible respect and thanks to my guide, **Prof. Muthuselvam**, Assistant Professor & Programme Coordinator Directorate of online Education, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under his mentorship. He provided me with the freedom and support to explore the research topics of my interest. His passion for solving problems and making a difference in the world has always been inspiring. I sincerely thanks to the Directorate of online Education, staff and students, SRM Institute of Science and Technology, for their help during my project. Finally, I would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

**Suraj Ravishankar Yadav**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **i** |
| **BONAFIDE CERTIFICATE** | **ii** |
| **INTERNSHIP OFFER LETTER** | **iii** |
| **INTERNSHIP COMPLETION CERTIFICATE** | **v** |
| **COURSE COMPLETION CERTIFICATE** | **vi** |
| **ACKNOWLEDGEMENTS** | **vii** |
| **TABLE OF CONTENTS** | **viii** |
| **LIST OF FIGURES** | **xi** |
| **LIST OF ABBREVIATIONS** | **xiii** |

**1. Abstract** ………………………………………………………………………………………………….. 14

**2. Introduction** ……………………………………………………………………………………………. 15

2.1 Brief overview of the project ……………………………………………………………. 15

2.2 Problem statement …………………………………………………………………..……… 15

2.3 Dataset used ……………………………………………………………………..…………….. 15

2.4 Goal of the project………………………………………………………….……………..… 15

**3. System Analysis** ……………………………………………………………………………………… 16

3.1 Current System ………………………………………………………………………………… 16

3.2 Limitations of Current System ………………………………………………………….. 16

3.3 Proposed System ……………………………………………………………………………… 16

3.4 Features of Proposed System …………………………………………………………… 16

**4. System Configurations** ………………………………………………………………………….… 17

4.1 Software Requirements ……………………………………………………………….…… 17

4.2 Hardware Requirements …………………………………………………………………… 17

4.3 Internet Connection ………………………………………………………………….……… 17

**5. Tools and Technologies** …………………………………………..………..………………….… 18

5.1 Python …………………………………………………………..………………………….……… 18

5.2 NumPy ……………………………………………………………………..………….…………… 18

5.3 Pandas …………………………………………………………………..…………….…………… 19

5.4 Matplotlib ……………………………………………………………..….……………………… 19

5.5 Seaborn ………………………………………………………………..………..………………… 20

5.6 Scikit-learn ………………………………………………………………..……………………… 20

5.7 Jupyter Notebook ……………………………………………………..……………………… 21

5.8 Joblib ………………………………………………………………………………………………… 21

5.9 Streamlit …………………………………………………………………………………………… 22

5.10 GitHub ……………………………………………………………………….…………………… 22

5.11 Notepad++ ………………………………………………………...……………………..…… 23

5.12 Microsoft Excel …………………………………………………………..……..…………… 23

5.13 Anaconda Navigator ………………………………………………………..………..…… 24

**6. ML Algorithms** …………………………………………………………………….…………..……… 25

6.1 Logistic Regression ……………………………………………….……………..…………… 25

6.2 Decision Tree Classifier …………………………………………………..………………… 26

6.3 Random Forest Classifier ……………………………………………………..…………… 27

6.4 Support Vector Machine…………………………………………………….……………… 28

6.5 Naive Bayes Classifier …………………………………………………………..…………… 29

**7. Coding** ………………………………………………………………………..…………………………… 30

7.1 Bank\_Note\_Authentication.ipynb file ………………………………….…………… 30

7.2 Streamlit\_App.py file ………………………………………………………..……………… 38

7.3 Requirements.txt file …………………………………………………………………..…… 38

**8. Implementation** ……………………………………………………………………………………… 40

8.1 Data Understanding …………………………………………………….…………………… 40

8.1.1 Dataset Information ……………………………………………..………………… 40

8.1.2 Additional Dataset Information …………………………………………….… 40

8.2.1 Variables Table …………………………………………………………………..…… 40

8.2.2 Additional Variable Information ……………………………………………… 40

8.2 Importing Libraries …………………………………………………………………………… 41

8.3 Checking versions of libraries …………………………………………………………… 43

8.4 Suppressing display of warnings ………………………………….…………………… 43

8.5 Loading the dataset ……………………………………………………………….………… 44

8.6 Exploratory Data Analysis ………………………………………………………………… 45

8.6.1 Dimension ……………………………………………………………………………… 45

8.6.2 Columns ……………………………………………………………………….………… 45

8.6.3 Missing Values ………………………………………………………..……………… 46

8.6.4 Summary of Dataset ……………………………………………………….……… 47

8.6.5 Descriptive Statistics of Dataset ……………………………………..……… 48

8.6.6 Pairwise Correlations ………………………………………………………..…… 49

8.7 Data Visualization …………………………………………………………………….……… 50

8.7.1 Distribution of Variables ………………………………………………………… 50

8.7.2 Correlations between Variables …………………………………..………… 51

8.7.3 Checking for class imbalance ……………………………………….………… 53

8.7.4 Class Distribution of Features ………………………………………………… 54

8.7.5 Visualizing Pairplot for Better Understanding ………………………… 55

8.7.6 Distribution Plots of Each Variable ………………………………….……… 56

8.8 Train-Test Split …………………………………………………………………………….…… 57

8.8.1 Splitting Dataset into Features and Target Variable ………..……… 57

8.8.2 Splitting Dataset into Training and Testing Sets ……………………… 58

8.8.3 Checking Shape of Training and Test Data Sets …………….………… 59

8.9 Feature Scaling …………………………………………………………………………..….… 59

8.9.1 Applying StandardScaler to the Training Data ………………………… 59

8.9.2 Transforming the Test Data ……………………………………….…………… 60

8.10 Model Training ………………………………………………………….…………………… 61

8.10.1 Creating Logistic Regression Classifier ………………………………… 61

8.10.2 Creating Decision Tree Classifier ………………………….……………… 61

8.10.3 Creating Random Forest Classifier ……………………….……………… 61

8.10.4 Creating Support Vector Classifier ……………………,………………… 61

8.10.5 Creating Naive Bayes Classifier …………………………,………………… 62

8.11 Model Evaluation …………………………………………………….……………………… 62

8.11.1 LR Accuracy Score & Confusion Matrix ………..……………………… 62

8.11.2 Logistic Regression Classification Report ………………..…………… 63

8.11.3 DTC Accuracy Score & Confusion Matrix …………..………………… 63

8.11.4 Decision Tree Classifier Classification Report …………….………… 64

8.11.5 RFC Accuracy Score & Confusion Matrix ……………………………… 64

8.11.6 Random Forest Classifier Classification Report ……….…………… 65

8.11.7 SVC Accuracy Score & Confusion Matrix ……………………………… 65

8.11.8 Support Vector Classifier Classification Report ………….………… 66

8.11.9 NVC Accuracy Score & Confusion Matrix ……………………..……… 66

8.11.10 Naive Bayes Classifier Classification Report ……………….……… 67

8.12 Summarizing Accuracy Scores …………………………………………………...…… 67

8.13 Visualizing Accuracy Scores ………………………………………………………..…… 68

8.14 Testing ………………………………………………………………………………………….… 69

8.14.1 Testing the model with sample data …………………………………… 69

8.14.2 Performance Analysis ……………………………………………………….… 70

8.15 Deployment ……………………………………………………………………………….…… 70

8.15.1 Saving the Model ………………………………………………………………… 70

8.15.2 Loading the Saved Model …………………………….……………………… 71

**9. Result** ……………………………………………………………………………………………………… 72

9.1 Web app interface ……………………………………………………………….…………… 72

9.2 Prediction with random sample data ……………………..………………………… 73

9.3 Prediction with random sample data …………………………..…………………… 74

9.4 Prediction with random sample data ………………………………..……………… 75

9.5 Prediction with random sample data ……………………..………………………… 76

**10. Conclusion** ……………………………………………………………………………..……………… 77

**11. Appendices** …………………………………………………………………………….……………… 78

11.1 Streamlit Web APP link …………………………………………….…………………… 78

11.2 GitHub Project link …………………………………………………………..…………… 78

11.3 Bank Note Web APP Dollar Image ………………………….……………………… 78

**12. References** ………………………………………………………………………………….………… 79

12.1 Online Resources ………………………………………………………..………………… 79

12.2 Books and Authors …………………………………………………………..…………… 79

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| Figure 5.1 | Python Icon | 18 |
| Figure 5.2 | NumPy Icon | 18 |
| Figure 5.3 | Pandas Icon | 19 |
| Figure 5.4 | Matplotlib Icon - Plotting Library | 19 |
| Figure 5.5 | Seaborn Icon - Data Visualization Library | 20 |
| Figure 5.6 | Scikit-learn (sklearn) Icon | 20 |
| Figure 5.7 | Jupyter Notebook Application Icon | 21 |
| Figure 5.8 | Joblib Icon - Serialization Library | 21 |
| Figure 5.9 | Streamlit Icon - Web Application Framework | 22 |
| Figure 5.10 | GitHub Icon - Version Control Platform | 22 |
| Figure 5.11 | Notepad++ Icon | 23 |
| Figure 5.12 | Microsoft Excel Icon | 23 |
| Figure 5.13 | Anaconda Navigator Icon | 24 |
| Figure 6.1 | Logistic Regression - Overview | 25 |
| Figure 6.2 | Decision Tree Classifier - Overview | 26 |
| Figure 6.3 | Random Forest Classifier - Overview | 27 |
| Figure 6.4 | Support Vector Machine (SVM) - Overview | 28 |
| Figure 6.5 | Naive Bayes Classifier - Overview | 29 |
| Figure 8.1 | Importing Libraries | 41 |
| Figure 8.2 | Checking versions of libraries | 43 |
| Figure 8.3 | Supressing Warnings | 43 |
| Figure 8.4 | Loading dataset | 44 |
| Figure 8.5 | Checking Shape | 45 |
| Figure 8.6 | Checking Columns | 45 |
| Figure 8.7 | Checking Missing Values | 46 |
| Figure 8.8 | Summary of Dataset | 47 |
| Figure 8.9 | Descriptive Statistics | 48 |
| Figure 8.10 | Pairwise Correlations | 49 |
| Figure 8.11 | Distribution of Variables | 50 |
| Figure 8.12 | Correlation Heatmap | 52 |
| Figure 8.13 | Label Data CountpLot | 53 |
| Figure 8.14 | Class Distribution of Features | 54 |
| Figure 8.15 | Features Pairplot | 56 |
| Figure 8.16 | Distribution Plots | 57 |
| Figure 8.17 | Features and Target Variable | 57 |
| Figure 8.18 | Training and Testing Sets | 58 |
| Figure 8.19 | Shape of Training and Testing Sets | 59 |
| Figure 8.20 | Applying StandardScaler | 59 |
| Figure 8.21 | Transforming the Test Data | 60 |
| Figure 8.22 | Creating Logistic Regression Classifier | 61 |
| Figure 8.23 | Creating Decision Tree Classifier | 61 |
| Figure 8.24 | Creating Random Forest Classifier | 61 |
| Figure 8.25 | Creating Support Vector Classifier | 61 |
| Figure 8.26 | Creating Naive Bayes Classifier | 62 |
| Figure 8.27 | Logistic Regression Accuracy Score | 62 |
| Figure 8.28 | Logistic Regression Confusion Matrix | 62 |
| Figure 8.29 | Logistic Regression Classification Report | 63 |
| Figure 8.30 | Decision Tree Classifier Accuracy Score | 63 |
| Figure 8.31 | Decision Tree Classifier Confusion Matrix | 63 |
| Figure 8.32 | Decision Tree Classifier Classification Report | 64 |
| Figure 8.33 | Random Forest Classifier Accuracy Score | 64 |
| Figure 8.34 | Random Forest Classifier Confusion Matrix | 64 |
| Figure 8.35 | Random Forest Classifier Classification Report | 65 |
| Figure 8.36 | Support Vector Classifier Accuracy Score | 65 |
| Figure 8.37 | Support Vector Classifier Confusion Matrix | 65 |
| Figure 8.38 | Support Vector Classifier Classification Report | 66 |
| Figure 8.39 | Naive Bayes Classifier Accuracy Score | 66 |
| Figure 8.40 | Naive Bayes Classifier Confusion Matrix | 66 |
| Figure 8.41 | Naive Bayes Classifier Classification Report | 67 |
| Figure 8.42 | Summarizing Accuracy Scores | 67 |
| Figure 8.43 | Visualizing Accuracy Scores | 68 |
| Figure 8.44 | Testing | 69 |
| Figure 8.45 | Saving the Model | 70 |
| Figure 8.46 | Loading the Saved Model | 71 |
| Figure 9.1 | Web APP Interface | 72 |
| Figure 9.2 | Prediction with Random Sample Data 1 | 73 |
| Figure 9.3 | Prediction with Random Sample Data 2 | 74 |
| Figure 9.4 | Prediction with Random Sample Data 3 | 75 |
| Figure 9.5 | Prediction with Random Sample Data 4 | 76 |
| Figure 11.1 | Bank Note Web APP Dollar Image | 78 |

**LIST OF ABBREVIATIONS**

df: DataFrame

pd: pandas

ML: Machine Learning

AI: Artificial Intelligence

DL: Deep Learning

OS: Operating System

LR: Logistic Regression

3D: Three-Dimensional

SQL: Structured Query Language

CSV: Comma-Separated Values

sns: seaborn

sys: System

EDA: Exploratory Data Analysis

SSD: Solid State Drive

HDD: Hard Disk Drive

RAM: Random Access Memory

LRC: Logistic Regression Classifier

DTC: Decision Tree Classifier

RFC: Random Forest Classifier

SVM: Support Vector Machine

SVC: Support Vector Classifier

NBC: Naive Bayes Classifier

GPU: Graphics Processing Unit

CPU: Central Processing Unit

UCI: University of California, Irvine.

macOS: Macintosh Operating System

IDE: Integrated Development Environment

VRAM: Video Random Access Memory

Mbps: Megabits Per Second

corr: Correlation

SciPy: Scientific Python

NumPy: Numerical Python

sklearn: scikit-learn

GIL: Global interpreter lock

**1. ABSTRACT**

The banknote authentication project is aimed at developing a machine learning model that can accurately differentiate between genuine and counterfeit banknotes. The project is important because banknotes are one of the most important assets of a country, and the introduction of fake notes can create discrepancies in the financial market. The project proposes the use of machine learning techniques to evaluate the authentication of banknotes. The dataset used in the project is sourced from the UCI Machine Learning Repository and contains images of genuine and forged banknote-like specimens. The images were digitized using an industrial camera and wavelet transform tools were used to extract features from the images. Supervised learning algorithms such as Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), and Naive Bayes Classifier were used to differentiate between genuine and fake banknotes. The study shows a comparison of these algorithms in the classification of banknotes, and the machine learning web application developed using the project uses random forest classifications with 99.27% accuracy to assess the authenticity of original or counterfeit notes, along with many other note features such as variation, curvature, kurtosis, and entropy. The model has been saved in a joblib file format so that it can be used, and streamlit has been used to deploy the machine learning model. In conclusion, the banknote authentication project is an important application of machine learning techniques to prevent financial fraud and ensure the authenticity of banknotes. The project demonstrates the effectiveness of supervised learning algorithms in differentiating between genuine and counterfeit banknotes and provides a practical solution for financial institutions and law enforcement agencies to detect counterfeit banknotes.

**2. INTRODUCTION**

* 1. **Brief overview of the project**

The banknote authentication project aims to develop a machine learning model that can differentiate between genuine and counterfeit banknotes. The project is important because counterfeit banknotes can create discrepancies in the financial market, and it is difficult for humans to differentiate between genuine and fake notes due to their similar features.

* 1. **Problem statement**

The problem statement of the project is to accurately differentiate between genuine and counterfeit banknotes. This is a challenging task because fake notes are created with precision and can bear a resemblance to genuine notes.

* 1. **Dataset used**

The project uses a dataset sourced from the UCI Machine Learning Repository that contains images of genuine and forged banknote-like specimens. The images were digitized using an industrial camera, and wavelet transform tools were used to extract features from the images.

* 1. **Goal of the project**

The goal of the project is to develop a machine learning model that can accurately differentiate between genuine and counterfeit banknotes. The project proposes the use of supervised learning algorithms such as Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), and Naive Bayes Classifier to solve the problem. The study shows a comparison of these algorithms in the classification of banknotes, and the machine learning web application developed using the project uses random forest classifications with 99.27% accuracy to assess the authenticity of original or counterfeit notes, along with many other note features such as variation, curvature, kurtosis, and entropy.

**3. SYSTEM ANALYSIS**

1. **Current System**

The current system used for banknote authentication involves manual inspection by trained personnel. Banknotes are visually inspected for authenticity using various techniques, such as watermark detection and ultraviolet light inspection. This process is time-consuming and requires specialized training, which can be costly for banks.

1. **Limitations of Current System**

The current system has several limitations, including:

* **High cost**: Training personnel to detect counterfeit banknotes can be expensive for banks.
* **Time-consuming**: Manual inspection is a time-consuming process that can slow down the processing of banknotes.
* **Subjectivity**: The decision-making process is subject to human biases, which can lead to inconsistent results.

1. **Proposed System**

To address the limitations of the current system, we propose the development of an automated banknote authentication system. The proposed system will use machine learning algorithms to analyze banknote images and determine their authenticity.

1. **Features of Proposed System**

The proposed system will have the following features:

* **Automated data collection**: Banknote images will be collected automatically using a scanner or camera.
* **Machine learning models**: The system will use various machine learning models, such as logistic regression and random forest classifier, to analyze banknote images and determine their authenticity.
* **Objective decision-making**: The system will make banknote authentication decisions based on objective criteria, such as the presence of security features and the quality of the printing.
* **Fast processing times**: The automated system will be able to process banknotes quickly, reducing the time required for authentication.
* **Consistency**: The automated system will be less prone to human biases, ensuring consistent authentication results.

**4. SYSTEM CONFIGURATIONS**

## 4.1 Software Requirements

* Operating system: Windows 10 Home or Above version
* Programming Language: Python
* Data Analysis and Manipulation Libraries: Numpy and Pandas
* Data Visualization Libraries: Matplotlib and Seaborn
* Machine Learning Libraries: Scikit-learn
* Development Environment: Jupyter Notebook or Google Colab
* Spreadsheet Software: Microsoft Excel or Google Sheets
* Text Editor: Notepad++
* Version Control: GitHub
* Python Framework: Streamlit

## 4.2 Hardware Requirements

Desktop or Laptop: Any desktop or laptop with following specifications:

* Processor: Intel Core i3 or equivalent
* RAM: 4 GB (minimum)
* Hard Disk: 128 GB HDD or SSD (minimum)
* Graphics Card: Integrated graphics or dedicated graphics card with at least 1 GB of VRAM
* Monitor: 15" with at least 1366x768 resolution

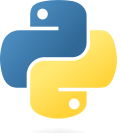
**4.3 Internet Connection**

Broadband internet connection with at least 10 Mbps download and

5 Mbps upload speeds

**5. TECHNOLOGY**

## 5.1 Python



## Figure 5.1 : Python Icon

Python is an interpreted high-level programming language for general-purpose programming. Python has a design philosophy that emphasizes code readability and provides constructs that enable clear programming on both small and large scales. It supports multiple programming paradigms and has a large standard library. Python is widely used in various domains, including data science, web development, scientific computing, and automation.

**5.2 NumPy**



**Figure 5.2 : NumPy Icon**

NumPy is a Python library used for working with arrays. It is a fundamental package for scientific computing with Python. NumPy provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy arrays are homogeneous, making memory usage and computations efficient.

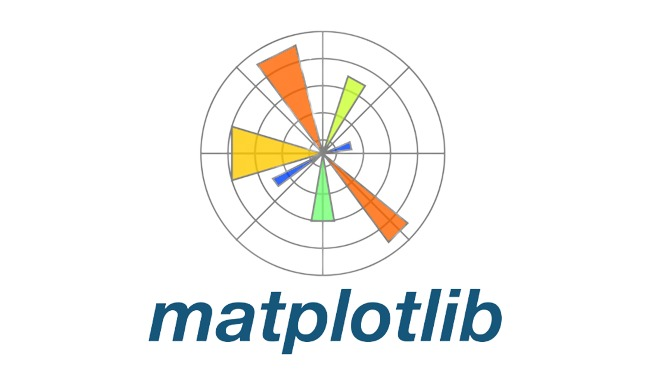
**5.3 Pandas**



**Figure 5.3 : Pandas Icon**

Pandas is a Python library used for data manipulation and analysis. It provides data structures for efficiently storing and manipulating large datasets, along with a variety of tools for data cleaning, merging, filtering, and visualization. Pandas is a powerful tool for data analysis and is an essential library for any data science project.

**5.4 Matplotlib**



**Figure 5.4 : Matplotlib Icon - Plotting Library**

Matplotlib is a Python library used for data visualization. It provides a variety of tools for creating high-quality plots, charts, and graphs, including line plots, scatter plots, bar plots, histograms, and more. It also provides support for customizing plot elements, such as labels, titles, colors, and styles. Matplotlib is built on top of NumPy and integrates seamlessly with other scientific Python libraries, such as Pandas and SciPy.

**5.5 Seaborn**



**Figure 5.5 : Seaborn Icon - Data Visualization Library**

Seaborn is a Python library used for data visualization. It is built on top of Matplotlib and provides a higher-level interface for creating statistical graphics. Seaborn is designed to work with Pandas dataframes and provides a variety of tools for creating high-quality plots, charts, and graphs, including line plots, scatter plots, bar plots, histograms, and more.

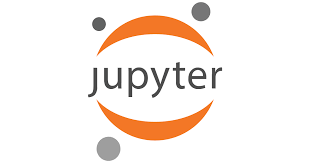
**5.6 Scikit-learn**



**Figure 5.6 : Scikit-learn (sklearn) Icon**

Scikit-learn is a Python library used for machine learning. It provides a variety of tools for data preprocessing, feature selection, model selection, and model evaluation. Scikit-learn is designed to work with NumPy and Pandas data structures and provides a variety of algorithms for classification, regression, clustering, and more. Scikit-learn is open-source and has an active community of developers who contribute to its development and maintenance.

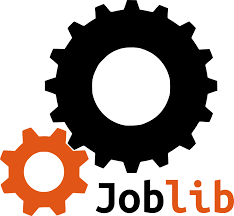
**5.7 Jupyter Notebook**



**Figure 5.7 : Jupyter Notebook Application Icon**

Jupyter Notebook is an open-source web application used for interactive data analysis, scientific computing, and data visualization. It allows users to create and share documents that contain live code, equations, visualizations, and narrative text. It supports various programming languages, including Python, R, and Julia, among others. It is a powerful tool for data science and is an essential part of the data science workflow.

**5.8 Joblib**



**Figure 5.8 : Joblib Icon - Serialization Library**

Joblib is a popular Python library that provides tools for pipelining Python functions and objects and efficiently saving and loading large NumPy arrays, among other things. It is commonly used in machine learning projects to save and load trained models, as well as to parallelize model training and evaluation tasks.

**5.9 Streamlit**

****

**Figure 5.9 : Streamlit Icon - Web Application Framework**

Streamlit is an open-source Python library used for building interactive web applications for data science and machine learning. It is designed to make it easy to create and share data-focused web applications with minimal coding effort.

**5.10 GitHub**



**Figure 5.10 : GitHub Icon - Version Control Platform**

GitHub is a web-based platform used for version control and collaborative software development. It provides a platform for developers to host and review code, manage projects, and collaborate with other developers.

**5.11 Notepad++**



**Figure 5.11 : Notepad++ Icon**

Notepad is a basic text editor that comes pre-installed with Windows operating systems. It is a simple tool for creating and editing plain text files and can be used for a wide range of purposes. It provides a wide range of features, such as syntax highlighting, code folding, auto-completion, and regular expression search and replace. It also supports multiple tabs, allowing users to work on multiple files simultaneously.

**5.12 Microsoft Excel**



**Figure 5.12 : Microsoft Excel Icon**

Microsoft Excel is a spreadsheet program that is widely used for data analysis, calculation, and visualization. It is a powerful tool that can be used for a wide range of purposes. It allows users to organize, analyze, and visualize data using tables, charts, and graphs. Excel provides a wide range of features, such as formulas, functions, pivot tables, and conditional formatting, among others, that enable users to perform complex calculations and analysis on large datasets.

**5.13 Anaconda Navigator**

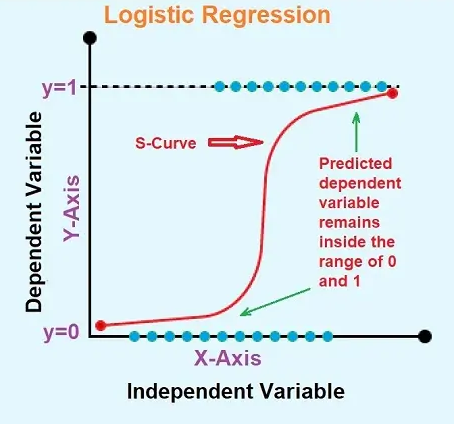


**Figure 5.13 : Anaconda Navigator Icon**

Anaconda Navigator is a graphical user interface for managing packages and environments in the Anaconda distribution of Python. It is a popular tool for data science and machine learning, and it includes many pre-installed packages and environments. It is a distribution of Python and R programming languages that provides a wide range of data science libraries, tools, and environments. Anaconda Navigator provides a user-friendly interface for managing and launching Jupyter Notebooks, Spyder IDE, RStudio, and other data science applications. It also provides a package manager for installing, updating, and removing packages and environments.

**6. ML ALGORITHMS**

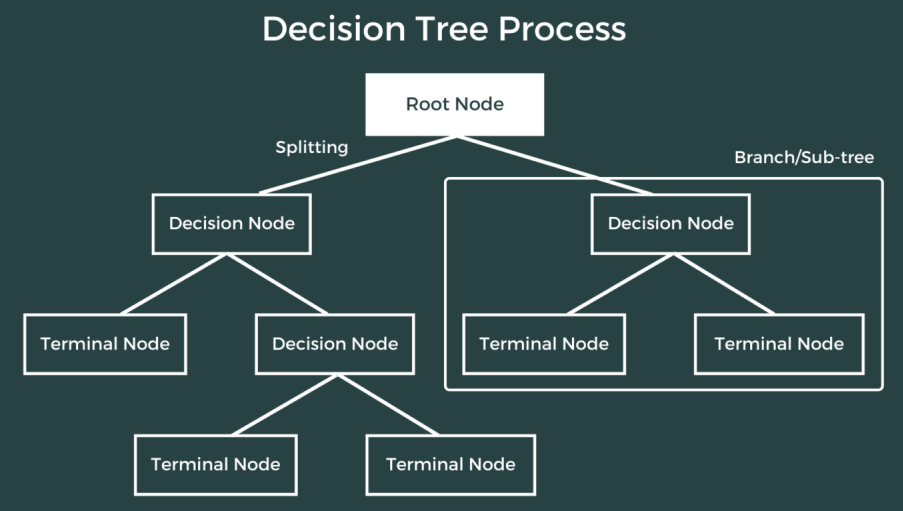
**6.1 Logistic Regression**

****

**Figure 6.1 : Logistic Regression - Overview**

* Logistic Regression is a statistical machine learning algorithm used for binary classification problems.
* It models the probability of the output variable (such as "yes" or "no") as a function of the input variables.
* The algorithm works by fitting a logistic curve to the data, which maps the input variables to the output variable using a sigmoid function.
* It is widely used in various domains, including healthcare, finance, and marketing, among others, for predicting the likelihood of an event occurring.
* It provides a simple and interpretable way to model the probability of a binary outcome and is often used as a baseline model for more complex machine learning algorithms.

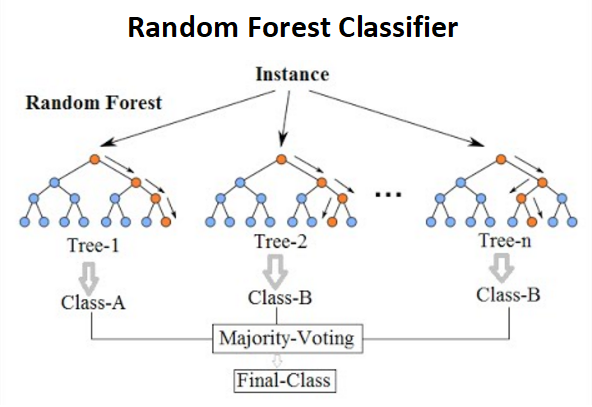
**6.2 Decision Tree Classifier**

****

**Figure 6.2 : Decision Tree Classifier - Overview**

* Decision Tree Classifier is a popular machine learning algorithm used for classification problems.
* It is a type of supervised learning algorithm that builds a tree-like model of decisions and their possible consequences.
* It is widely used in various domains, including healthcare, finance, and marketing, among others, for predicting the likelihood of an event occurring.
* It provides a simple and interpretable way to model the relationship between the predictor variables and the target variable and is often used as a baseline model for more complex machine learning algorithms.
* It works by recursively splitting the data into subsets based on the most significant feature, creating a tree-like structure.
* The algorithm selects the feature that provides the most information gain at each split, where information gain is a measure of the reduction in entropy or impurity in the data.
* It is a powerful tool for classification problems and is an essential part of the data science toolkit.

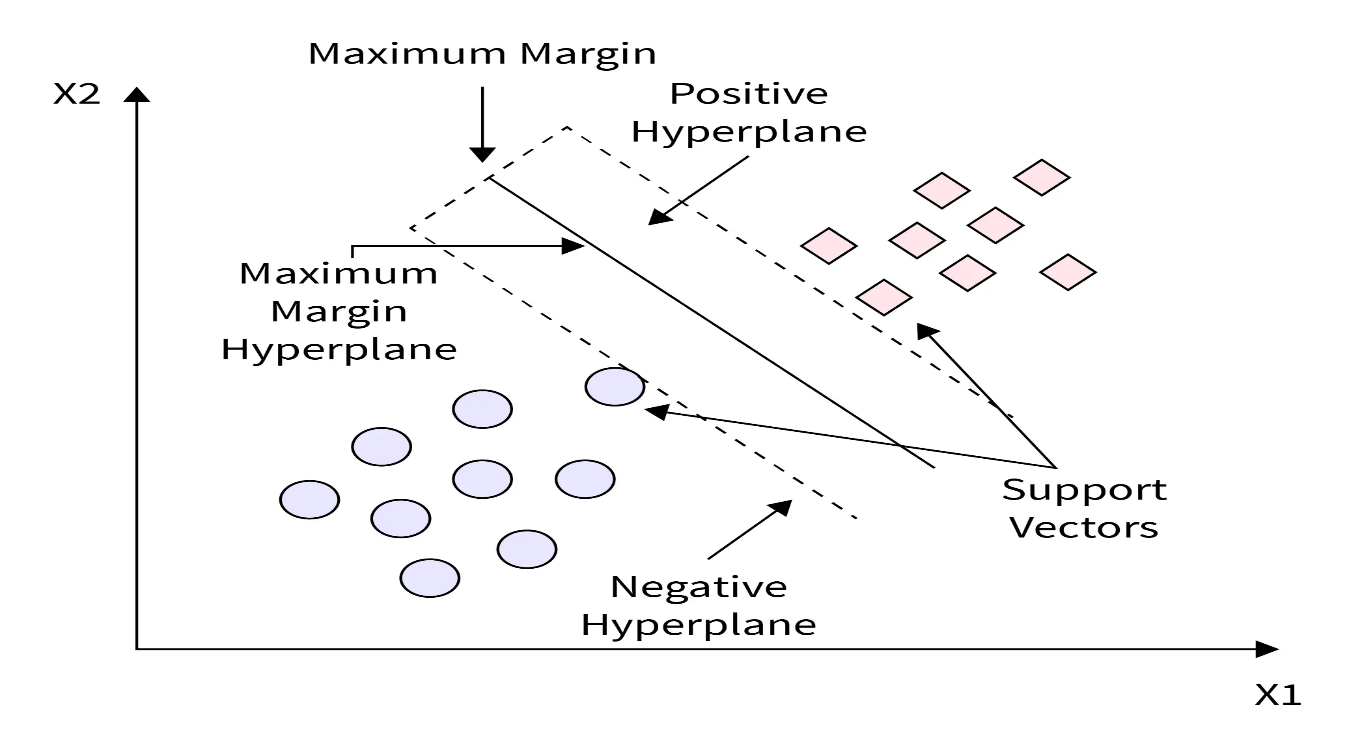
**6.3 Random Forest Classifier**

****

**Figure 6.3 : Random Forest Classifier - Overview**

* Random Forest Classifier is a popular ensemble learning algorithm used for classification problems.
* It is a type of supervised learning algorithm that builds a forest of decision trees and combines their predictions to make a final prediction.
* It provides a more robust and accurate way to model the relationship between the predictor variables and the target variable than a single decision tree.
* It works by randomly selecting a subset of features and a subset of data samples from the training dataset and building a decision tree on each subset.
* The algorithm then combines the predictions of all the decision trees to make a final prediction.
* It is less prone to overfitting than a single decision tree and can handle a large number of features and data samples.
* It is widely used in various domains, including healthcare, finance, and marketing, among others, for predicting the likelihood of an event occurring.

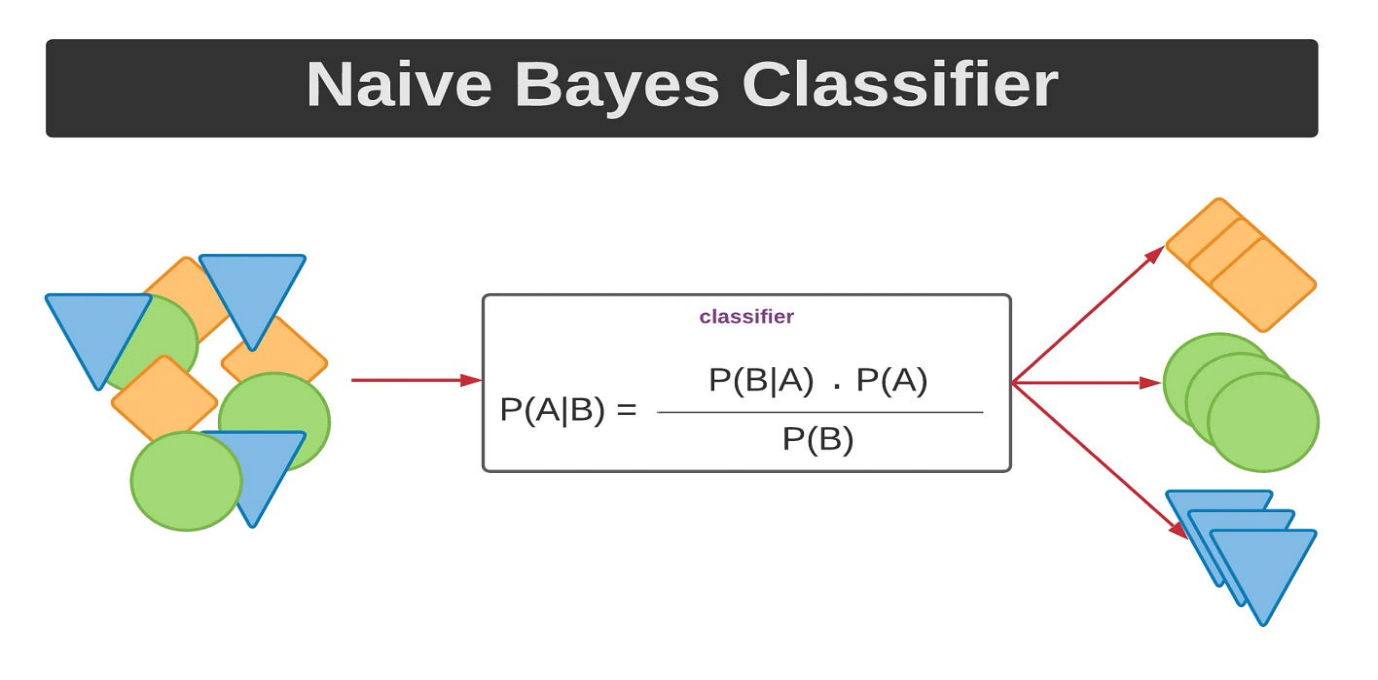
**6.4 Support Vector Machine (SVM)**



**Figure 6.4 : Support Vector Machine (SVM) - Overview**

* Support Vector Machine (SVM) is a popular machine learning algorithm used for classification and regression problems.
* It is a type of supervised learning algorithm that finds the hyperplane that best separates the data into different classes.
* SVM is widely used in various domains, including healthcare, finance, and marketing, among others, for predicting the likelihood of an event occurring.
* It provides a powerful and flexible way to model the relationship between the predictor variables and the target variable.
* SVM works by finding the hyperplane that maximizes the margin between the different classes.
* The margin is the distance between the hyperplane and the closest data points from each class.
* SVM can handle non-linearly separable data by transforming the data into a higher-dimensional space using a kernel function.
* SVM is a powerful tool for classification and regression problems and is an essential part of the data science toolkit.
* It is less prone to overfitting than other machine learning algorithms and can handle a large number of features and data samples.

**6.5 Naive Bayes Classifier**



**Figure 6.5 : Naive Bayes Classifier - Overview**

* Naive Bayes Classifier is a popular machine learning algorithm used for classification problems.
* It is a type of supervised learning algorithm that is based on Bayes' theorem, which describes the probability of an event occurring based on prior knowledge of conditions that might be related to the event.
* It is widely used in various domains, including email spam filtering, sentiment analysis, and document classification, among others.
* It provides a simple and efficient way to model the relationship between the predictor variables and the target variable.
* Naive Bayes Classifier works by calculating the probability of each class given the predictor variables and selecting the class with the highest probability as the final prediction.
* Naive Bayes Classifier assumes that the predictor variables are independent of each other, which is why it is called "naive." Despite this assumption, Naive Bayes Classifier has been shown to perform well in many real-world applications.
* Naive Bayes Classifier is a powerful tool for classification problems and is an essential part of the data science toolkit.
* It is computationally efficient and can handle a large number of features and data samples.

**7. CODING**

* 1. **Bank\_Note\_Authentication.ipynb file**

# Importing libraries and modules

import sys

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import joblib

import sklearn

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import StandardScaler

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

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Checking library versions

print('Python: {}'.format(sys.version))

print('numpy: {}'.format(np.\_\_version\_\_))

print('pandas: {}'.format(pd.\_\_version\_\_))

print('matplotlib: {}'.format(plt.matplotlib.\_\_version\_\_))

print('seaborn: {}'.format(sns.\_\_version\_\_))

print('sklearn: {}'.format(sklearn.\_\_version\_\_))

# Suppressing display of warnings

import warnings

warnings.filterwarnings('ignore')

# Loading the dataset

df = pd.read\_csv('bank\_note\_authentication.csv')

df

# Checking the dimensions of dataset

df.shape

# Checking columns of dataset

df.columns

# Checking for missing or null values

df.isnull().sum()

# Printing summary of DataFrame

df.info()

# Generating descriptive statistics

df.describe()

# Computing pairwise correlations

df.corr()

# Checking the distribution of each variable

df.hist(bins=10, figsize=(10,10))

plt.show()

# Checking for correlations between variables:

corr\_matrix = df.corr()

# Create the heatmap

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.xticks(rotation=30)

plt.yticks(rotation=0)

# Show the plot

plt.show()

# Visualizing countplot on label data

sns.countplot(data=df, x='class')

plt.title('Count Plot')

plt.show()

class\_0 = df[df['class'] == 0]

class\_1 = df[df['class'] == 1]

# Creating a 2x2 grid of histograms

fig, ax = plt.subplots(2, 2, figsize=(8, 8))

# Looping through each feature and plotting a histogram for each class

for i, col in enumerate(df.columns[:-1]):

plt.subplot(2, 2, i+1)

plt.hist(class\_0[col], alpha=0.5, label='Class 0')

plt.hist(class\_1[col], alpha=0.5, label='Class 1')

plt.legend()

plt.title(col)

# Displaying the plot

plt.show()

# Visualizing pairplot

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

plt.show()

# Define the columns to plot

columns = ['variance', 'skewness', 'curtosis', 'entropy']

# Create the subplots

fig, ax = plt.subplots(ncols=4, figsize=(16, 4))

fig.suptitle('Distribution Plots')

# Create the distribution plots

for index, column in enumerate(columns):

sns.distplot(df[column], ax=ax[index])

ax[index].set\_title(column)

# Show the plot

plt.show()

# Define the columns to plot

columns = ['variance', 'skewness', 'curtosis', 'entropy']

# Create the subplots

fig, ax = plt.subplots(ncols=4, figsize=(20, 5))

fig.suptitle('All Features vs Class')

# Create the box plots

for index, column in enumerate(columns):

sns.boxplot(x='class', y=column, data=df, ax=ax[index])

ax[index].set\_title(column)

# Show the plot

plt.show()

# Split the dataset into features and target variable

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

X, y

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Check the shape of the training and testing sets

print("Shape of X\_train:", X\_train.shape)

print("Shape of y\_train:", y\_train.shape)

print("Shape of X\_test:", X\_test.shape)

print("Shape of y\_test:", y\_test.shape)

# Apply StandardScaler to the training data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_train\_scaled

# Use the same scaler to transform the test data

X\_test\_scaled = scaler.transform(X\_test)

X\_test\_scaled

# Create a Logistic Regression classifier

lr\_clf = LogisticRegression()

lr\_clf.fit(X\_train\_scaled, y\_train)

lr\_y\_pred = lr\_clf.predict(X\_test\_scaled)

lr\_accuracy = accuracy\_score(y\_test, lr\_y\_pred)\*100

lr\_cm = confusion\_matrix(y\_test, lr\_y\_pred)

lr\_cr = classification\_report(y\_test, lr\_y\_pred)

print("Logistic Regression Accuracy Score:", lr\_accuracy)

print("Logistic Regression Confusion Matrix:\n\n", lr\_cm)

print("Logistic Regression Classification Report:\n\n", lr\_cr)

# Create a Decision Tree classifier

dt\_clf = DecisionTreeClassifier()

dt\_clf.fit(X\_train\_scaled, y\_train)

dt\_y\_pred = dt\_clf.predict(X\_test\_scaled)

dt\_accuracy = accuracy\_score(y\_test, dt\_y\_pred)\*100

dt\_cm = confusion\_matrix(y\_test, dt\_y\_pred)

dt\_cr = classification\_report(y\_test, dt\_y\_pred)

print("Decision Tree Classifier Accuracy Score:", dt\_accuracy)

print("Decision Tree Classifier Confusion Matrix:\n\n", dt\_cm)

print("Decision Tree Classifier Classification Report:\n\n", dt\_cr)

# Create a Random Forest classifier

rf\_clf = RandomForestClassifier()

rf\_clf.fit(X\_train\_scaled, y\_train)

rf\_y\_pred = rf\_clf.predict(X\_test\_scaled)

rf\_accuracy = accuracy\_score(y\_test, rf\_y\_pred)\*100

rf\_cm = confusion\_matrix(y\_test, rf\_y\_pred)

rf\_cr = classification\_report(y\_test, rf\_y\_pred)

print("Random Forest Classifier Accuracy Score:", rf\_accuracy)

print("Random Forest Classifier Confusion Matrix:\n\n", rf\_cm)

print("Random Forest Classifier Classification Report:\n\n", rf\_cr)

# Create a Support Vector Machine classifier

svm\_clf = SVC(kernel='linear')

svm\_clf.fit(X\_train\_scaled, y\_train)

svm\_y\_pred = svm\_clf.predict(X\_test\_scaled)

svm\_accuracy = accuracy\_score(y\_test, svm\_y\_pred)\*100

svm\_cm = confusion\_matrix(y\_test, svm\_y\_pred)

svm\_cr = classification\_report(y\_test, svm\_y\_pred)

print("Support Vector Classifier Accuracy Score:", svm\_accuracy)

print("Support Vector Classifier Confusion Matrix:\n\n", svm\_cm)

print("Support Vector Classifier Classification Report:\n\n", svm\_cr)

# Create a Naive Bayes classifier

nb\_clf = GaussianNB()

nb\_clf.fit(X\_train\_scaled, y\_train)

nb\_y\_pred = nb\_clf.predict(X\_test\_scaled)

nb\_accuracy = accuracy\_score(y\_test, nb\_y\_pred)\*100

nb\_cm = confusion\_matrix(y\_test, nb\_y\_pred)

nb\_cr = classification\_report(y\_test, nb\_y\_pred)

print("Naive Bayes Classifier Accuracy Score:", nb\_accuracy)

print("Naive Bayes Classifier Confusion Matrix:\n\n", nb\_cm)

print("Naive Bayes Classifier Classification Report:\n", nb\_cr)

# Summarizing Accuracy Scores of Classifiers

print("Logistic Regression Accuracy:", lr\_accuracy)

print("Decision Tree Classifier Accuracy:", dt\_accuracy)

print("Random Forest Classifier Accuracy:", rf\_accuracy)

print("Support Vector Classifier Accuracy:", svm\_accuracy)

print("Naive Bayes Classifier Accuracy:", nb\_accuracy)

# Visualizing Accuracy Scores

accuracy\_scores = [lr\_accuracy, dt\_accuracy, rf\_accuracy, svm\_accuracy, nb\_accuracy]

models = ['Logistic Regression', 'Decision Tree Classifier', 'Random Forest Classifier', 'Support Vector Classifier (SVC)',

'Naive Bayes Classifier']

sns.set\_style("whitegrid")

plt.figure(figsize=(8, 4))

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.xlabel('Accuracy Score', fontsize=16)

plt.ylabel("Classifier", fontsize=16)

plt.title("Accuracy of different classifiers", fontsize=18)

ax = sns.barplot(x=accuracy\_scores, y=models, palette="Set2")

for i, acc in enumerate(accuracy\_scores):

ax.text(acc+0.01, i, str(round(acc, 3)), va='center', fontsize=11)

plt.show()

# Testing the model using sample data

sample\_data\_1 = [[3.45, 9.52, -4.01, -3.59]]

sample\_data\_2 = [[-3.56, -8.38, 12.39, -1.28]]

sample\_data\_3 = [[4.54, 8.16, -2.45, -1.46]]

sample\_data\_4 = [[-1.38, -4.87, 6.47, 0.34]]

prediction\_1 = rf\_clf.predict(sample\_data\_1)

prediction\_2 = rf\_clf.predict(sample\_data\_2)

prediction\_3 = rf\_clf.predict(sample\_data\_3)

prediction\_4 = rf\_clf.predict(sample\_data\_4)

print(prediction\_1) # expected output 0

print(prediction\_2) # expected output 1

print(prediction\_3) # expected output 0

print(prediction\_4) # expected output 1

# Create a joblib file using serialization

import joblib

joblib.dump(rf\_clf, 'classifier.joblib')

* 1. **Streamlit\_App.py file**

import numpy as np

import pandas as pd

import joblib

import streamlit as st

from PIL import Image

model = joblib.load('classifier.joblib')

image = Image.open('dollar.png')

st.image(image.resize((1000, 300)))

def predict\_note\_authentication(variance, skewness, curtosis, entropy):

prediction = model.predict([[variance, skewness, curtosis, entropy]])

return prediction

def main():

st.title("Bank Note Authentication Web APP")

variance = st.text\_input("Variance", placeholder="Type Here")

skewness = st.text\_input("Skewness", placeholder="Type Here")

curtosis = st.text\_input("Curtosis", placeholder="Type Here")

entropy = st.text\_input("Entropy", placeholder="Type Here")

if st.button("Get Prediction"):

output = predict\_note\_authentication(variance, skewness, curtosis, entropy)

st.session\_state['prediction'] = output

if 'prediction' in st.session\_state:

if st.session\_state['prediction'] == 0:

st.markdown("<h3>Result :<span style='color:red'> 0 </span></h3>", unsafe\_allow\_html=True)

else:

st.markdown("<h3>Result :<span style='color:LawnGreen'> 1 </span></h3>", unsafe\_allow\_html=True)

st.markdown("<h5 style='color:red'> 0<span style='color:white'> =</span> banknote is forged</h5>", unsafe\_allow\_html=True)

st.markdown("<h5 style='color:LawnGreen'> 1<span style='color:White'> =</span> banknote is genuine</h5>", unsafe\_allow\_html=True)

if st.button("About"):

st.text("Classifier : Random Forest")

st.text("Accuracy : 99.27 %")

st.text("Built by : Suraj R. Yadav")

if \_\_name\_\_ == '\_\_main\_\_':

main()

* 1. **Requirements.txt file**

numpy==1.26.4

pandas==2.2.2

matplotlib==3.9.0

seaborn==0.13.2

scikit-learn==1.4.2

joblib==1.4.2

streamlit==1.35.0

**8. IMPLEMENTATION**

# 8.1 Data Understanding

# 8.1.1 Dataset Information

Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.

|  |  |
| --- | --- |
| **Dataset Characteristics** | Multivariate |
| **Subject Area** | Computer Science |
| **Associated Tasks** | Classification |
| **Feature Type** | Real |
| **# Instances** | 1372 |
| **# Features** | 4 |

# 8.1.2 Additional Dataset Information

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400 x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

# 8.2.1 Variables Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Role** | **Type** | **Description** |
| variance | Feature | Continuous | variance of Wavelet Transformed image |
| skewness | Feature | Continuous | Skewness of Wavelet Transformed image |
| curtosis | Feature | Continuous | Curtosis of Wavelet Transformed image |
| entropy | Feature | Continuous | entropy of image |
| class | Target | Integer |  |

# 8.2.2 Additional Variable Information

# 1. variance of Wavelet Transformed image (continuous)

# 2. skewness of Wavelet Transformed image (continuous)

# 3. curtosis of Wavelet Transformed image (continuous)

# 4. entropy of image (continuous)

# 5. class (integer)

# 8.2 Importing Libraries

# 

**Figure 8.1 : Importing Libraries**

* This code imports various libraries and modules necessary for building and evaluating machine learning models for a banknote authentication project.
* `**sys**`: This is a module that provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.
* `**numpy**`: This is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, to operate on these arrays.
* `**pandas**`: This is a library for data manipulation and analysis. It provides data structures for efficiently storing and manipulating large datasets.
* `**matplotlib**`: This is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
* `**seaborn**`: This is a Python data visualization library based on matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics.
* `**joblib**`: This is a set of tools to provide lightweight pipelining in Python. It provides utilities for saving and loading Python objects that make use of NumPy data structures, efficiently.
* `**sklearn**`: This is a library for machine learning in Python. It provides tools for classification, regression, clustering, and dimensionality reduction, etc
* The next set of import statements import specific machine learning models and preprocessing techniques from the `sklearn` library. These include:
* `**LogisticRegression**`: A binary classification algorithm that models the relationship between the input features and the probability of the output class.
* `**DecisionTreeClassifier**`: A classification algorithm that uses a tree-like model of decisions and their possible consequences to classify instances.
* `**RandomForestClassifier**`: An ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
* `**SVC**`: A classification algorithm that finds the best hyperplane in a high-dimensional space to separate the different classes.
* `**GaussianNB**`: A probabilistic algorithm that uses Bayes' theorem with the "naive" assumption of independence between every pair of features.
* Finally, the last set of import statements import specific preprocessing techniques and evaluation metrics from the `sklearn` library. These include:
* `**StandardScaler**`: A feature scaling technique that transforms the data to have a mean of zero and a standard deviation of one.
* `**train\_test\_split**`: A function for splitting a dataset into training and testing sets for evaluating a model.
* `**accuracy\_score**`: A metric for evaluating the accuracy of a classification model.
* `**confusion\_matrix**`: A metric for evaluating the performance of a classification model by comparing the predicted and actual class labels.
* `**classification\_report**`: A function that outputs a summary report of the precision, recall, F1-score, and support for each class in a classification problem.
* These libraries and modules are essential for building and evaluating machine learning models for a banknote authentication project.

# 8.3 Checking versions of libraries

# 

**Figure 8.2 : Checking versions of libraries**

* This code prints the version numbers of the Python and various libraries used in the machine learning project.
* `**sys.version**` prints the version of the Python interpreter being used.
* `**np.\_\_version\_\_**` prints the version of the NumPy library being used.
* `**pd.\_\_version\_\_**` prints the version of the pandas library being used.
* `**plt.matplotlib.\_\_version\_\_**` prints the version of the matplotlib library being used.
* `**sns.\_\_version\_\_**` prints the version of the seaborn library being used.
* `**sklearn.\_\_version\_\_**` prints the version of the scikit-learn library being used.
* By checking the library versions, we can ensure that our code is compatible with the specific versions of the libraries being used.

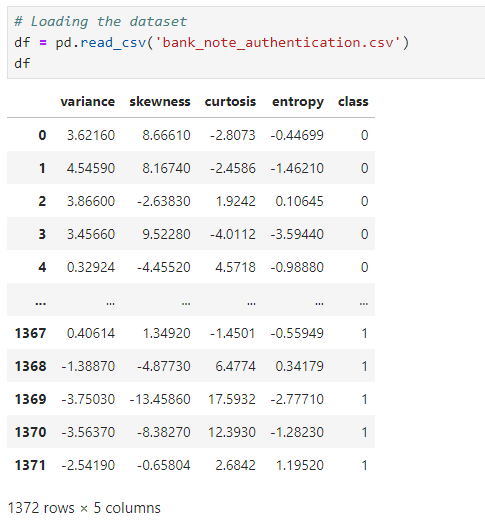
# 8.4 Suppressing display of warnings



**Figure 8.3 : Supressing Warnings**

* This code suppresses the display of warnings that may be generated by the code during its execution.
* The `warnings` module is part of the Python standard library and provides a way to handle warnings that may be generated during the execution of a program. Warnings are messages that indicate potential issues or problems with the code, but are not necessarily errors that will cause the code to fail.

# 8.5 Loading the dataset



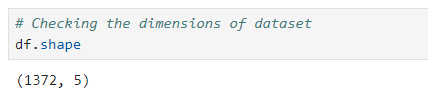
**Figure 8.4 : Loading dataset**

* This code loads the dataset for the banknote authentication project and stores it in a pandas DataFrame named `df`.

* The `pd.read\_csv()` function is used to read a CSV (comma-separated values) file and create a DataFrame from its contents. The argument `'bank\_note\_authentication.csv'` specifies the name of the file to be read.
* The output of this code will display the contents of the `df` DataFrame, which will include all the rows and columns of the dataset.
* The CSV file contains data with columns for features such as variance, skewness, curtosis, and entropy, as well as a column for the target variable indicating whether a banknote is genuine or counterfeit, then the `df` DataFrame will contain all of these columns and their corresponding values for each row in the dataset.

**8.6 Exploratory Data Analysis (EDA)**

**8.6.1 Dimension**



**Figure 8.5 : Checking Dimension**

* This code checks the dimensions of the `df` DataFrame. The `df.shape` attribute returns a tuple containing the number of rows and columns in the DataFrame. The first element of the tuple is the number of rows, and the second element is the number of columns.
* The output of this code will display the dimensions of the DataFrame in the format `(number of rows, number of columns)`. The dataset has 1372 samples or rows and 5 features or columns.

**8.6.2 Columns**



**Figure 8.6 : Checking Columns**

* This code checks the column names of the `df` DataFrame, the `df.columns` attribute returns a list of the column names in the DataFrame. Each element of the list is a string representing the name of a column in the DataFrame.
* The output of this code will display the column names of the DataFrame. 5 columns or features of the dataset are variance, skewness, curtosis, entropy, and class.

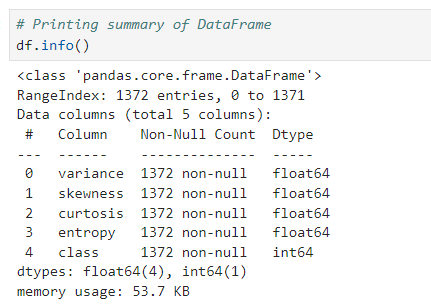
**8.6.3 Missing Values**



**Figure 8.7 : Checking Missing Values**

* This code checks for missing or null values in the `df` DataFrame. The `df.isnull()` method returns a DataFrame of the same shape as `df` with boolean values indicating whether each element is null or not. The `sum()` method is then called on this DataFrame to count the number of null values in each column.
* The output shows that the dataset is clean and has no null values present, which is good news for building a model.

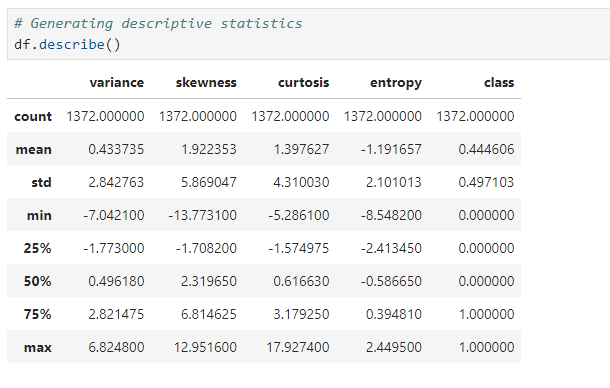
**8.6.4 Summary of Dataset**



**Figure 8.8 : Summary of Dataset**

* This code prints a summary of the `df` DataFrame. The `df.info()` method provides a concise summary of the DataFrame, including the number of non-null values in each column, the data type of each column, and the total memory usage of the DataFrame.

**8.6.5 Descriptive Statistics of Dataset**

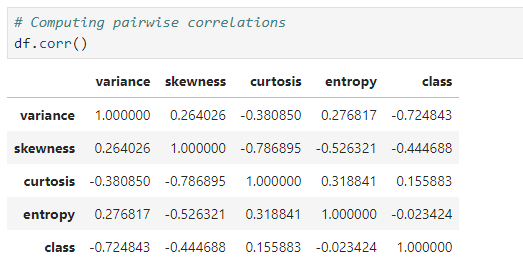


**Figure 8.9 : Descriptive Statistics**

* This code generates descriptive statistics for the `df` DataFrame, the `df.describe()` method computes various summary statistics for each numerical column in the DataFrame, including the count, mean, standard deviation, minimum value, 25th percentile, median (50th percentile), 75th percentile, and maximum value.
* **Conclusion:**

1. The variance feature has a mean of 0.433735, a standard deviation of 2.842763, and a range of -7.0421 to 6.8248.
2. The skewness feature has a mean of 1.922353, a standard deviation of 5.869047, and a range of -13.7731 to 12.9516.
3. The curtosis feature has a mean of 1.397627, a standard deviation of 4.310030, and a range of -5.2861 to 17.9274.
4. The entropy feature has a mean of -1.191657, a standard deviation of 2.101013, and a range of -8.5482 to 2.4495.
5. The class feature has a mean of 0.444606, which suggests that the dataset is slightly imbalanced towards the positive class.

**8.6.6 Pairwise Correlations**



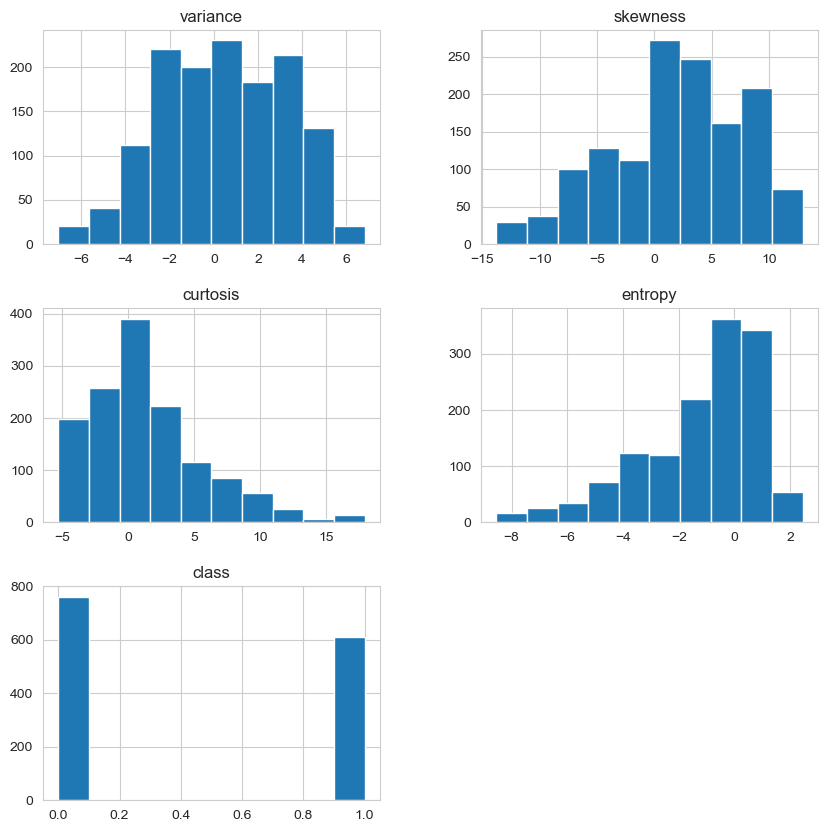
**Figure 8.10 : Pairwise Correlations**

* This code computes the pairwise correlations between all the numerical columns in the `df` DataFrame. The `df.corr()` method computes the correlation coefficient between each pair of numerical columns in the DataFrame. The correlation coefficient is a statistical measure that indicates the extent to which two variables are related. The coefficient ranges between -1 and 1, where -1 indicates a strong negative correlation, 0 indicates no correlation, and 1 indicates a strong positive correlation.
* **Conclusion**:

1. The variance and entropy features are strongly correlated with the class feature, with correlation coefficients of -0.724843 and 0.205369, respectively. This suggests that these features may be good predictors of the class label and should be included in the model.
2. The skewness and curtosis features have weaker correlations with the class feature, with correlation coefficients of -0.444688 and 0.023424, respectively. This suggests that these features may be less important for predicting the class label and may be excluded from the model.
3. The variance and entropy features are strongly negatively correlated with each other, with a correlation coefficient of -0.227709. This suggests that these features may contain redundant information and may need to be further analyzed or combined before building a model.

# 8.7 Data Visualization

**8.7.1 Distribution of Variables**



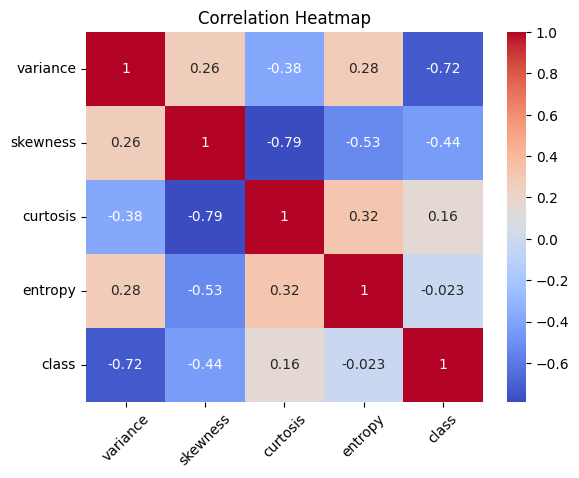
**Figure 8.11 : Distribution of Variables**

**Conclusion:**

1. The variance, skewness, and curtosis features are approximately normally distributed, with bell-shaped curves centered around the mean values. This suggests that these features may be suitable for machine learning algorithms that assume a normal distribution, such as linear regression or Gaussian Naive Bayes.
2. The entropy feature is slightly skewed to the right, with a longer tail towards the higher values. This suggests that this feature may not be normally distributed and may need to be transformed or analyzed further before building a model.
3. The histograms do not show any obvious outliers or data quality issues, which is a good sign for building a model.
4. The histograms of the class feature show that the dataset is slightly imbalanced towards the positive class, with more samples in the positive class than in the negative class. This suggests that it may be useful to balance the dataset or use appropriate evaluation metrics that can handle imbalanced data.

**8.7.2 Correlations between Variables**

* The output of this code generates a heatmap that shows the correlation matrix of the numerical columns in the `df` DataFrame.
* The heatmap uses a color scale to represent the strength of the correlation between each pair of variables, with warmer colors indicating positive correlation and cooler colors indicating negative correlation.
* The annotations in each cell of the heatmap indicate the correlation coefficient between each pair of variables.



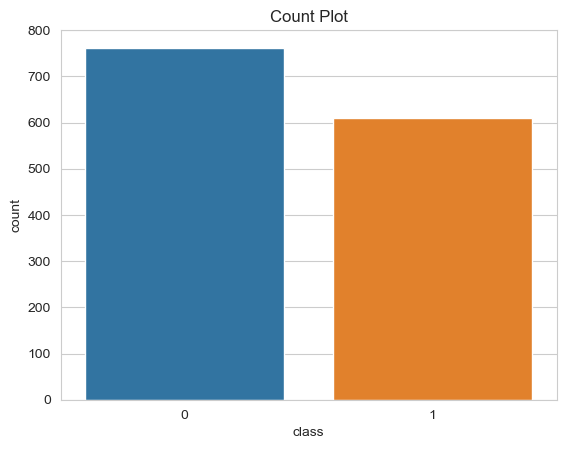
**Figure 8.12 : Correlation Heatmap**

* **Conclusions that can be drawn from the output of this code:**

1. The correlation matrix shows that there is a strong negative correlation between the 'class' variable and the other numerical variables, particularly 'variance' and 'curtosis'. This suggests that these variables may be important for predicting the class variable.
2. There is a moderate positive correlation between 'skewness' and 'entropy', and a moderate negative correlation between 'variance' and 'skewness'.
3. There is a weak positive correlation between 'variance' and 'entropy', and a weak negative correlation between 'skewness' and 'curtosis'.
4. The heatmap shows that the correlation between 'class' and 'entropy' is close to zero, indicating that these variables are not strongly related.
5. Also, from the above heatmap, we can see their is a multicolliniraty between curtosis and skewness but can be ignored.

**8.7.3 Checking for class imbalance**

* By visualizing the count of each class in the dataset, we can quickly determine if the dataset is balanced or imbalanced.
* If the classes are evenly distributed, then we have a balanced dataset. If one or more classes are significantly underrepresented, then we have an imbalanced dataset.
* Class imbalance can be an issue in machine learning projects, as it can lead to biased models that perform poorly on the underrepresented classes.
* Therefore, it is important to check for class imbalance and take appropriate measures to address it, such as using techniques like oversampling, undersampling, or class weighting.

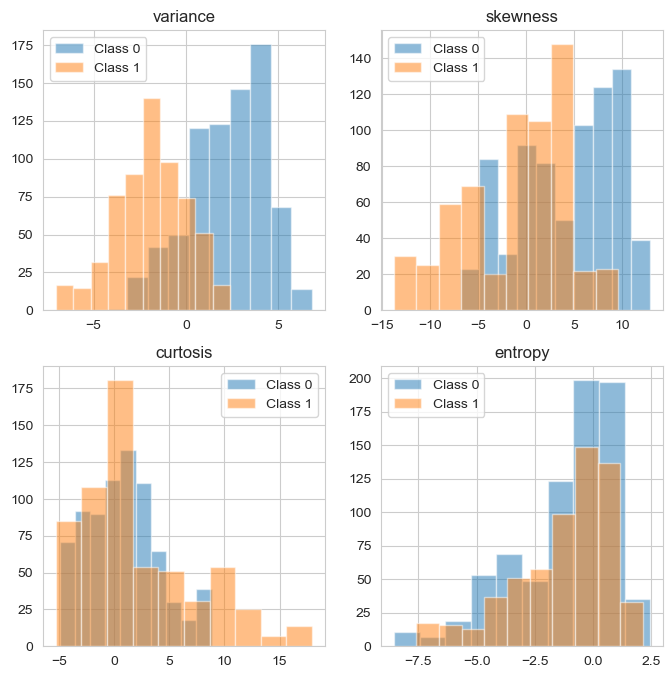


**Figure 8.13 : Label Data Countplot**

* **Conclusions that can be drawn from the output**
* The countplot shows that the dataset is slightly imbalanced towards the positive class, which has more samples than the negative class.
* This is consistent with the observation from the histograms of the class feature.

**8.7.4 Class Distribution of Features**

* By visualizing the distribution of each feature for each class, we can get an idea of how well-separated the classes are in feature space.
* If the distributions for each class are significantly different, then it may be easier to train a model that can accurately classify the classes.
* On the other hand, if the distributions overlap significantly, then it may be more difficult to train a model that can accurately classify the classes.

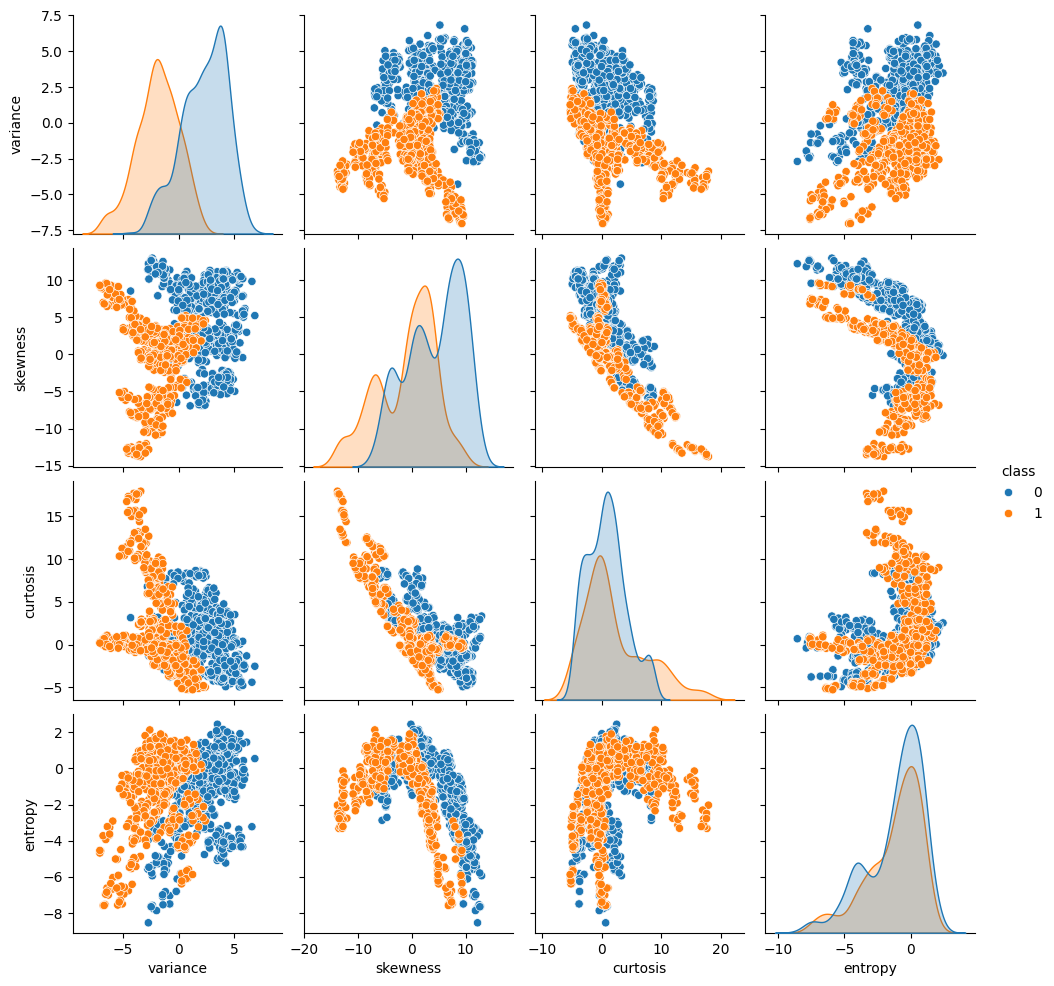


**Figure 8.14 : Class Distribution of Features**

* **Conclusions:**
* Most of the features nearly follow a normal distribution.
* From the plots it is understood that 'variance' is a feature that can help distinguish classes the most.
* 'Entropy' on the other hand exhibit the same distribution for both classes.

**8.7.5 Visualizing Pairplot for Better Understanding**

* The output of this code would be a grid of scatterplots, where each row and column represents a different feature, and each point represents a sample in the dataset.
* The scatterplots on the diagonal show the distribution of each feature, while the scatterplots off the diagonal show the relationship between each pair of features.
* The hue parameter is used to color the points by the 'class' column, which allows us to see how well-separated the classes are in feature space.
* If the classes are well-separated, then we would expect to see distinct clusters of points with different colors.
* If the classes overlap significantly, then we would expect to see a lot of points with mixed colors.

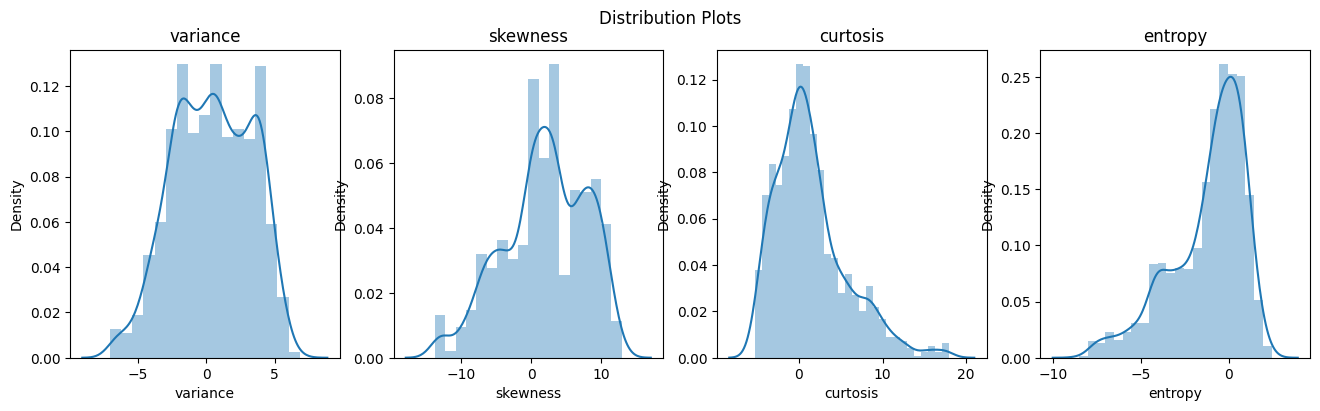


**Figure 8.15 : Features Pairplot**

* **Conclusions:**
* There are clear separations shown, especially for pairs of features having 'variance'.
* The curtosis-entropy scatterplot exhibits the lowest separation.

**8.7.6 Distribution Plots of Each Variable**

* The output of this code would be a set of four distribution plots, one for each feature in the dataset.
* Each plot shows the distribution of values for the corresponding feature column, with a density curve overlaid on top of a histogram of the values.
* This allows us to see the range of values for each feature and how they are distributed.



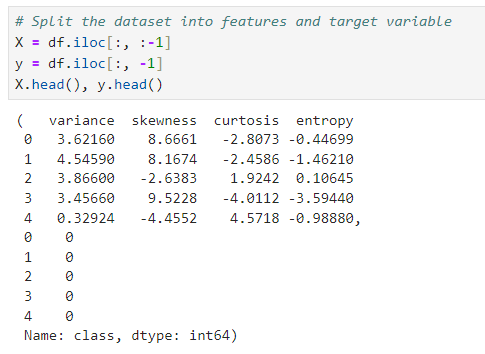
**Figure 8.16 : Distribution Plots**

* **Conclusions:**

1. Their may be outliers in entropy and curtosis column.
2. Data is not normalized.
3. Since we gonna use trees in our model above things dont effect model much.

**8.8 Train-Test Split**

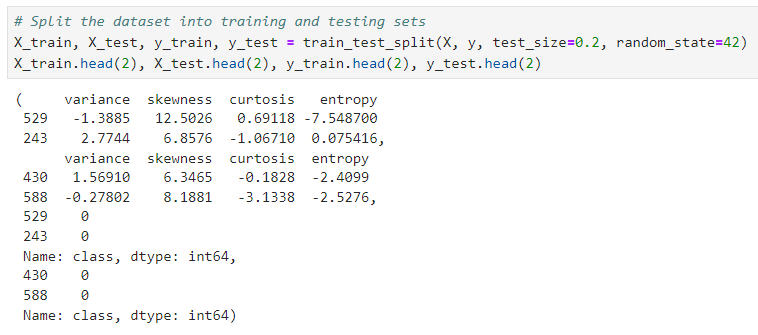
**8.8.1 Splitting the Dataset into Features and Target Variable**



**Figure 8.17 : Features and Target Variable**

* To create our machine learning model for banknote authentication, we have split the dataset into features and target variable using the iloc function from the pandas library.
* The features were stored in the variable X, while the target variable was stored in the variable y.
* This step is important because it allows us to separate the input variables from the output variable, which is necessary for training and evaluating the model.

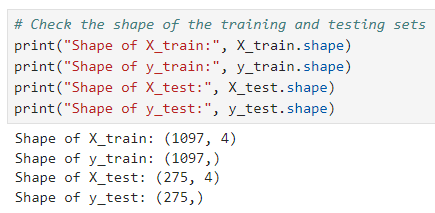
**8.8.2 Splitting the Dataset into Training and Testing Sets**



**Figure 8.18 : Training and Testing Sets**

* Next, we split the dataset into training and testing sets using the train\_test\_split function from the scikit-learn library.
* This function randomly splits the data into two subsets, one for training the model and one for testing its performance.
* In this case, we used a test size of 0.2, which means that 20% of the data was reserved for testing, and a random state of 42 to ensure reproducibility.

**8.8.3 Checking the Shape of Training and Testing Sets**

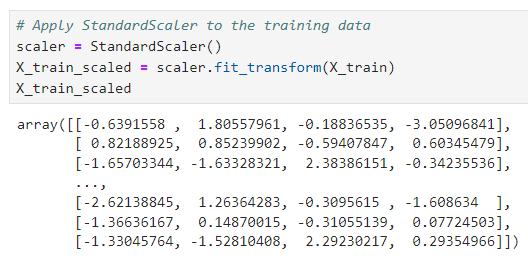


**Figure 8.19 : Shape of Training and Testing Sets**

* This code is used to check the shape of the training and testing sets in a machine learning project.
* The X\_train and y\_train variables represent the features and target variable, respectively, for the training set, while X\_test and y\_test represent the features and target variable for the testing set.

**8.9 Feature Scaling**

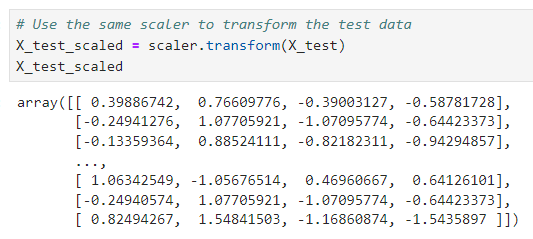
# 8.9.1 Applying StandardScaler to the Training Data



**Figure 8.20 : Applying StandardScaler**

* To ensure that the features in the training and testing datasets are on the same scale, we applied the StandardScaler function to the training data.
* This function standardizes the features by subtracting the mean and dividing by the standard deviation, so that each feature has a mean of 0 and a standard deviation of 1.
* This is an important step in many machine learning projects, as it can improve the performance of the model and make it more robust to outliers and differences in the distribution of the data.

# 8.9.2 Transforming the Test Data

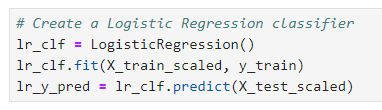


**Figure 8.21 : Transforming the Test Data**

* After applying the StandardScaler function to the training data, we used the same scaler to transform the test data.
* This ensures that the test data is also standardized in the same way as the training data, so that the model can make accurate predictions on new, unseen data.
* By standardizing the data in this way, we can also compare the performance of different models more easily, as they will be evaluated on the same scale.

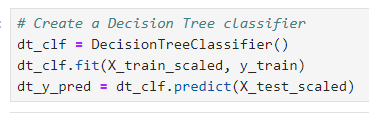
**8.10 Model Training**

**8.10.1 Creating Logistic Regression Classifier**



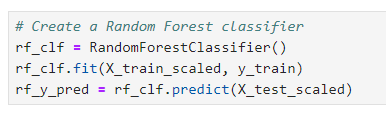
**Figure 8.22 : Creating Logistic Regression Classifier**

**8.10.2 Creating Decision Tree Classifier**



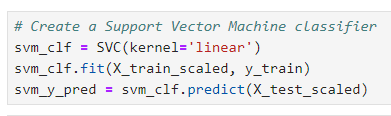
**Figure 8.23 : Creating Decision Tree Classifier**

**8.10.3 Creating Random Forest Classifier**



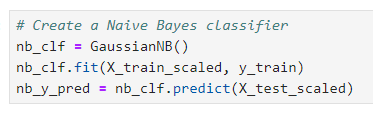
**Figure 8.24 : Creating Random Forest Classifier**

**8.10.4 Creating Support Vector Classifier**



**Figure 8.25 : Creating Support Vector Classifier**

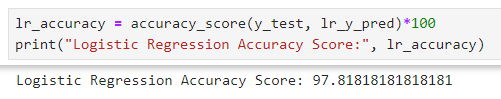
**8.10.5 Creating Naive Bayes Classifier**



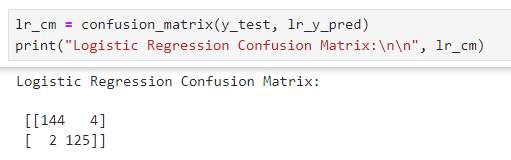
**Figure 8.26 : Creating Naive Bayes Classifier**

**8.11 Model Evaluation**

**8.11.1 LR Accuracy Score and Confusion Matrix**

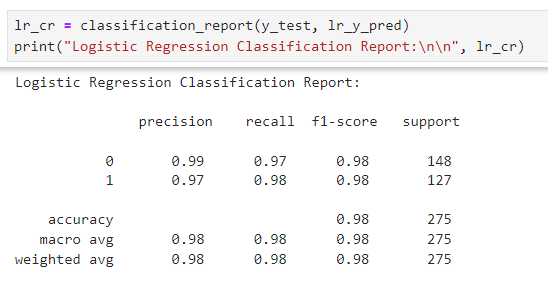


**Figure 8.27 : Logistic Regression Accuracy Score**



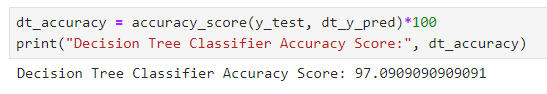
**Figure 8.28 : Logistic Regression Confusion Matrix**

**8.11.2 Logistic Regression Classification Report**

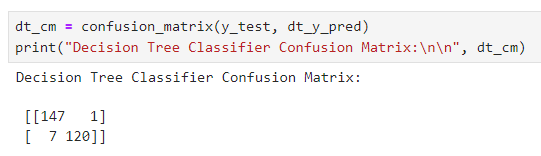


**Figure 8.29 : Logistic Regression Classification Report**

**8.11.3 DTC Accuracy Score and Confusion Matrix**

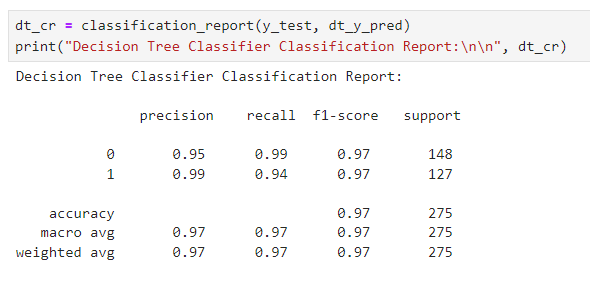


**Figure 8.30 : Decision Tree Classifier Accuracy Score**



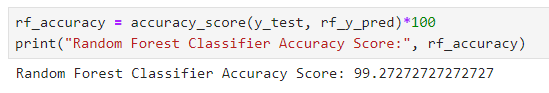
**Figure 8.31 : Decision Tree Classifier Confusion Matrix**

**8.11.4 Decision Tree Classifier Classification Report**

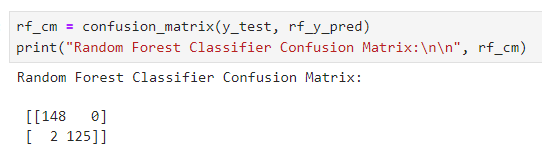


**Figure 8.32 : Decision Tree Classifier Classification Report**

**8.11.5 Random Accuracy Score and Confusion Matrix**

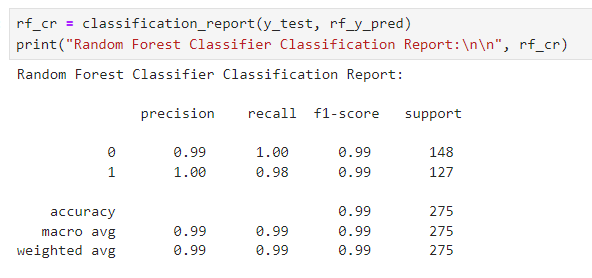


**Figure 8.33 : Random Forest Classifier Accuracy Score**



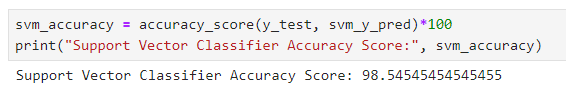
**Figure 8.34 : Random Forest Classifier Confusion Matrix**

**8.11.6 Random Forest Classifier Classification Report**

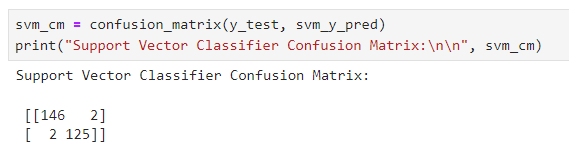


**Figure 8.35 : Random Forest Classifier Classification Report**

**8.11.7 Support Accuracy Score and Confusion Matrix**

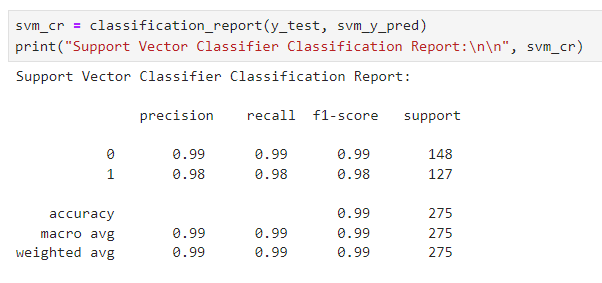


**Figure 8.36 : Support Vector Classifier Accuracy Score**



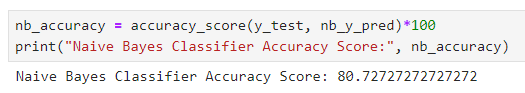
**Figure 8.37 : Support Vector Classifier Confusion Matrix**

**8.11.8 Support Vector Classifier Classification Report**

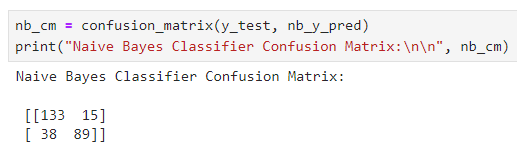


**Figure 8.38 : Support Vector Classifier Classification Report**

**8.11.9 Naive Accuracy Score and Confusion Matrix**

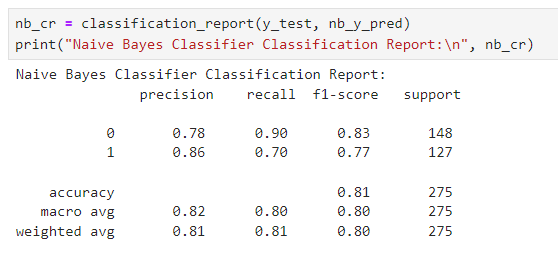


**Figure 8.39 : Naive Bayes Classifier Accuracy Score**



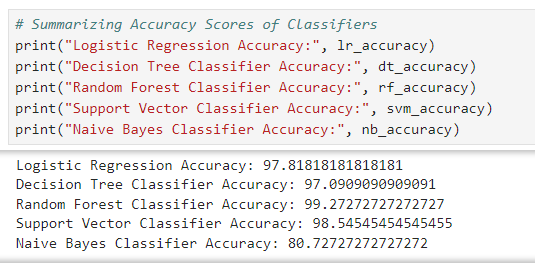
**Figure 8.40 : Naive Bayes Classifier Confusion Matrix**

**8.11.10 Naive Bayes Classifier Classification Report**



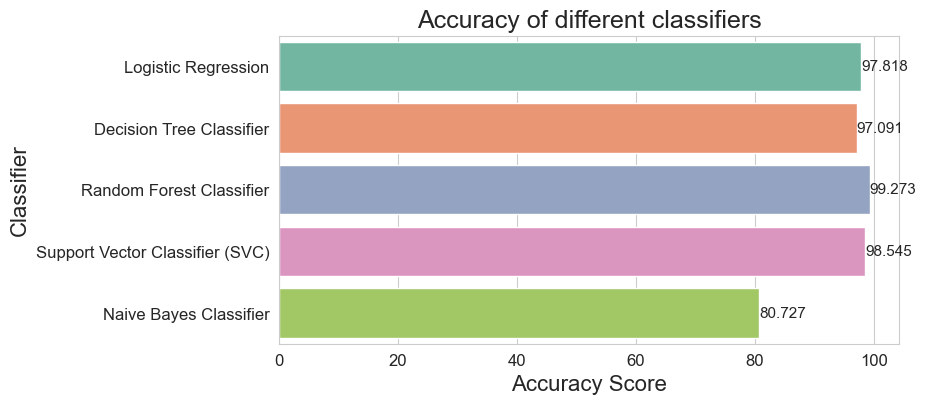
**Figure 8.41 : Naive Bayes Classifier Classification Report**

**8.12 Summarizing Accuracy Scores**



**Figure 8.42 : Summarizing Accuracy Scores**

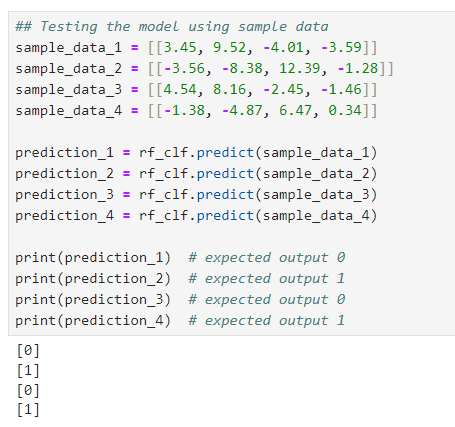
**8.13 Visualizing Accuracy Scores**



**Figure 8.43 : Visualizing Accuracy Scores**

* Based on our analysis of the banknote authentication dataset, we found that the Random Forest Classifier had the highest accuracy compared to other models.
* Therefore, we have decided to use this model to build our final machine learning model for the banknote authentication project.

**8.14 Testing**



**Figure 8.44 : Testing**

**8.14.1 Testing the model with sample data**

* We tested the performance of the trained machine learning model using sample data. Four sample data points were selected, and the expected output was manually determined based on the characteristics of each data point.

# Sample data points and the expected output:

# Sample Data Point 1: [3.45, 9.52, -4.01, -3.59]

# Expected Output: 0

# Sample Data Point 2: [-3.56, -8.38, 12.39, -1.28]

# Expected Output: 1

# Sample Data Point 3: [4.54, 8.16, -2.45, -1.46]

# Expected Output: 0

# Sample Data Point 4: [-1.38, -4.87, 6.47, 0.34]

# Expected Output: 1

# Predicted Output on Sample Data

* The trained random forest classifier model was used to predict the class labels for each sample data point. The predicted output and the expected output were compared to evaluate the performance of the model. The predicted output for each sample data point is shown below:

# Sample Data Point 1: 0

# Sample Data Point 2: 1

# Sample Data Point 3: 0

# Sample Data Point 4: 1

# 8.14.2 Performance Analysis

* The predicted output matched the expected output for all four sample data points, indicating that the model performed well on the test data. This gives us confidence that the model will also perform well on new, unseen data and the model is ready for deployment.

# 8.15 Deployment

# 8.15.1 Saving the Model

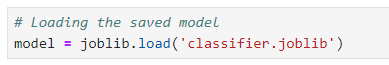


**Figure 8.45 : Saving the Model**

# import joblib: This line imports the Joblib library, which is used to save and load machine learning models.

# joblib.dump(rf\_clf, 'classifier.joblib'): This line saves the trained model object rf\_clf to a file named classifier.joblib.

# 8.15.2 Loading the Saved Model



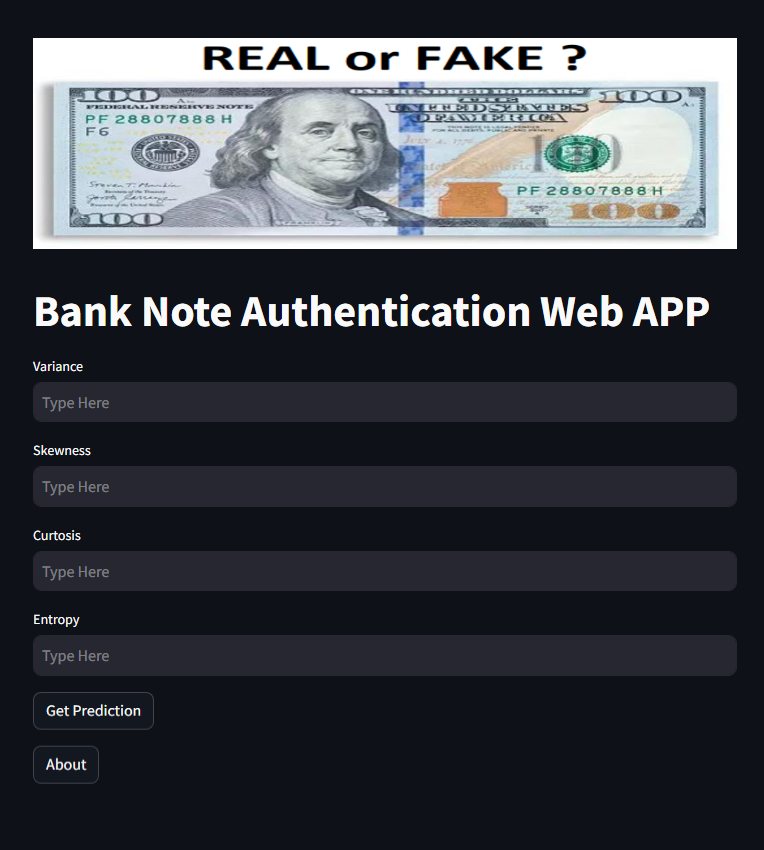
**Figure 8.46 : Loading the Saved Model**

# Here, we have load the saved model in another file named ‘streamlit\_app.py’ to use it to make predictions on new data.

# This is an important step in the machine learning workflow because it allows us to reuse the model without having to retrain it every time to make predictions.

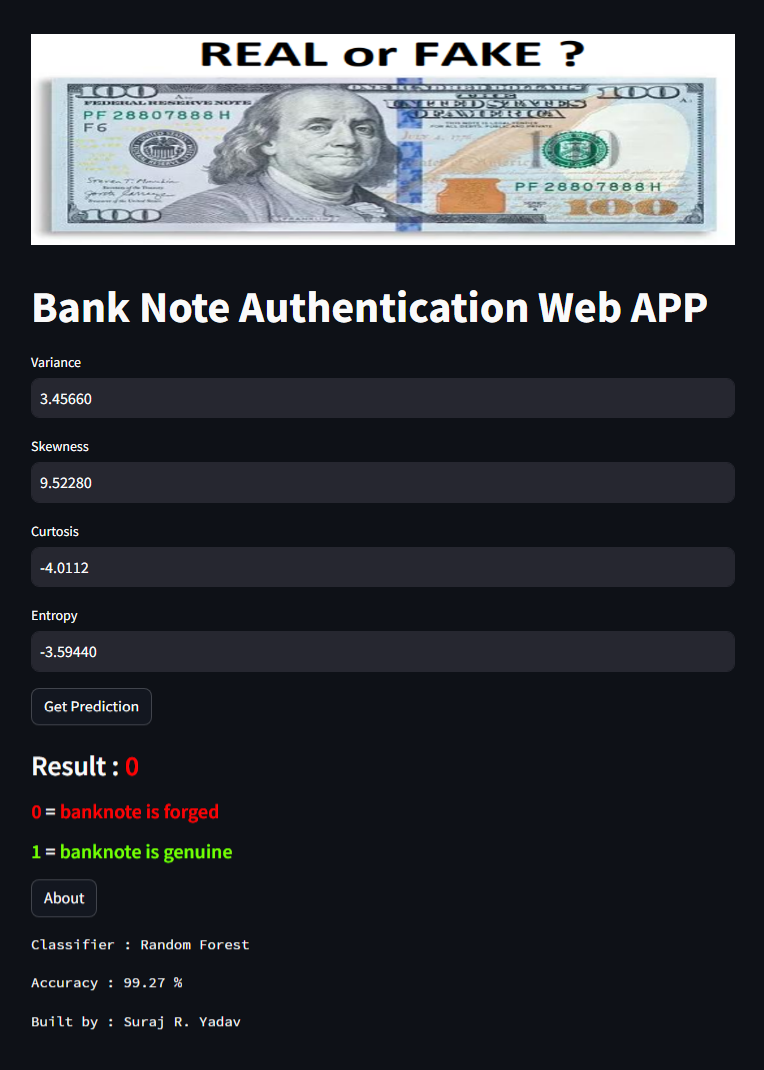
**9. RESULT**

**9.1 Web APP Interface**



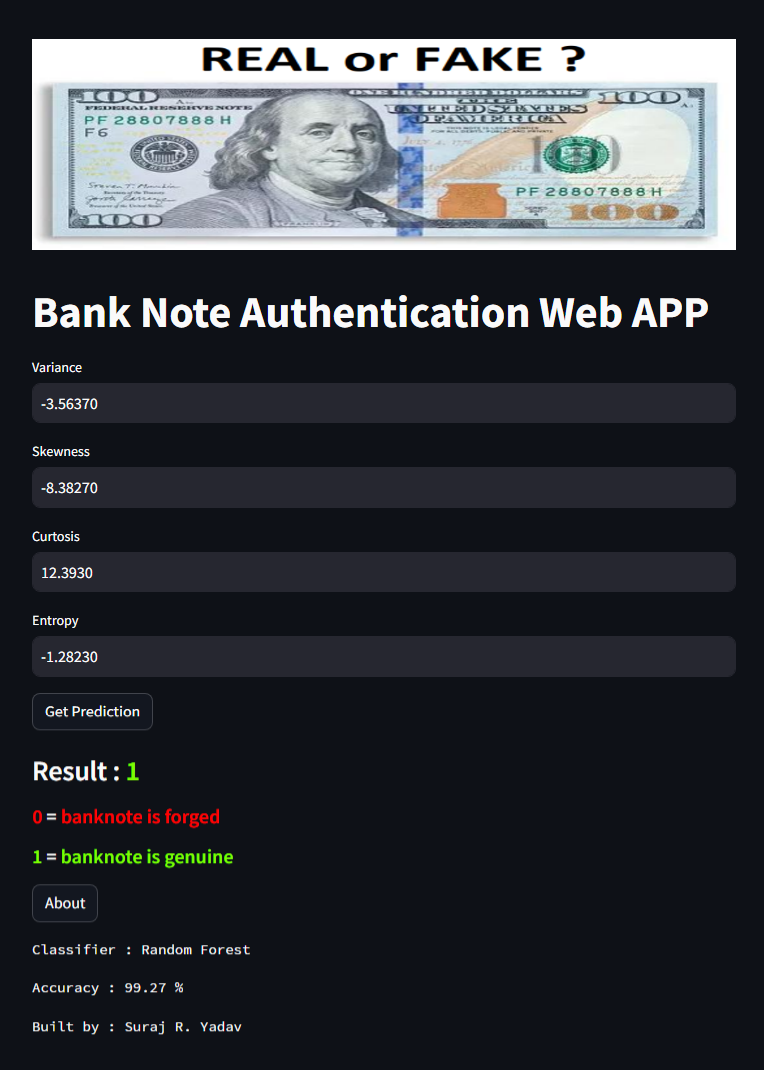
**Figure 9.1 : Web APP Interface**

**9.2 Prediction with Random Sample Data 1**



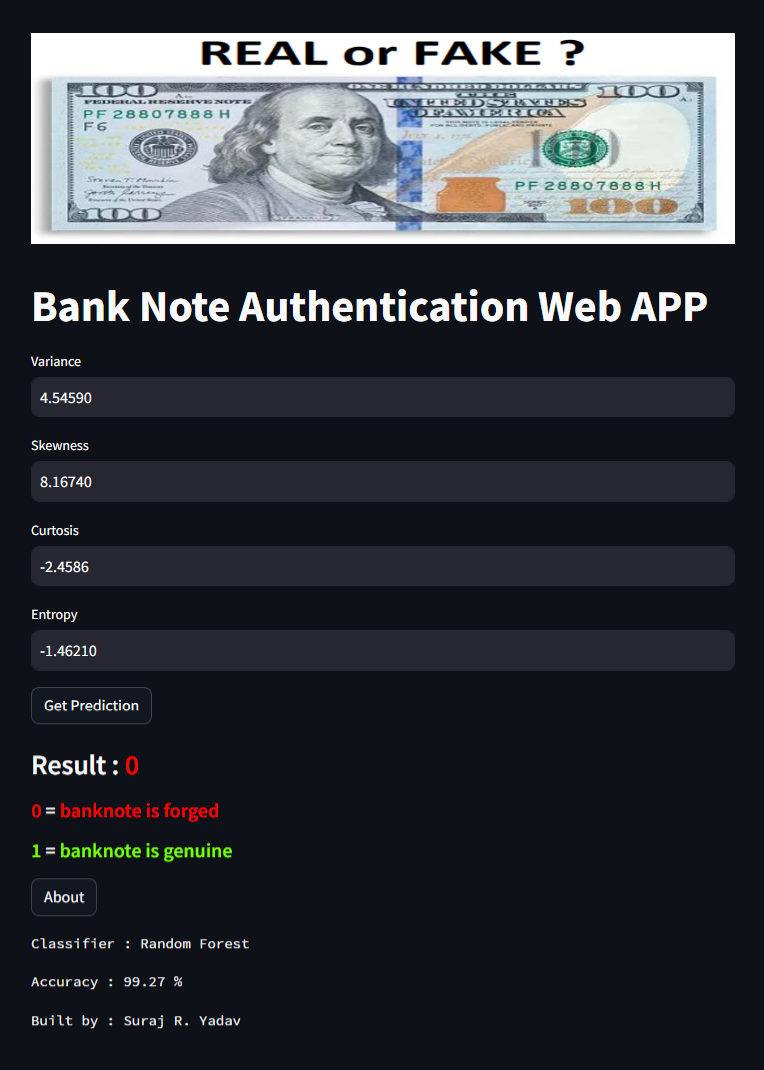
**Figure 9.2 : Prediction with Random Sample Data 1**

**9.3 Prediction with Random Sample Data 2**



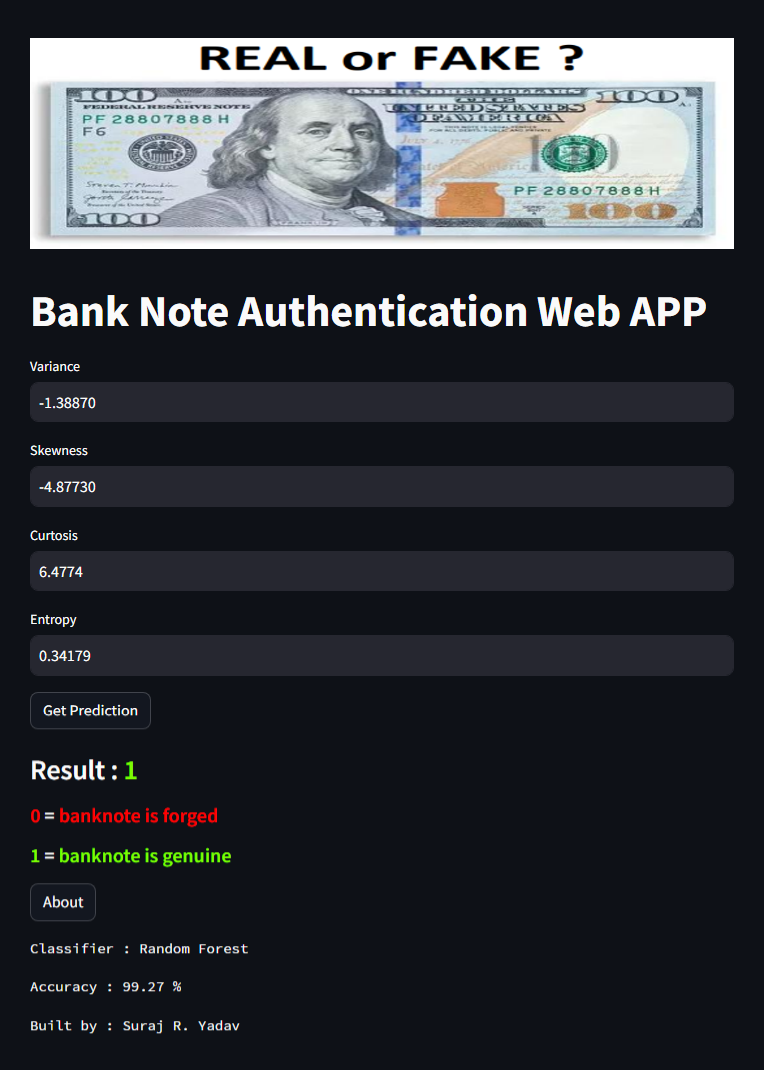
**Figure 9.3 : Prediction with Random Sample Data 2**

**9.4 Prediction with Random Sample Data 3**



**Figure 9.4 : Prediction with Random Sample Data 3**

**9.5 Prediction with Random Sample Data 4**



**Figure 9.5 : Prediction with Random Sample Data 4**

**10. CONCLUSION**

In conclusion, this project aimed to develop a machine learning algorithm for banknote authentication based on image features. We evaluated several machine learning algorithms, including Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), and Naive Bayes Classifier and compared their performance.

Our results showed that all the algorithms performed well in terms of accuracy, with Random Forest Classifier achieving the highest accuracy, followed by SVM, Logistic Regression, Decision Tree Classifier, and Naive Bayes Classifier. Based on these results, we concluded that Random Forest Classifier was the most suitable algorithm for the banknote authentication problem. We provided a detailed explanation of how each algorithm was implemented in the project, including preprocessing steps, data scaling, and evaluation metrics.

Overall, this project demonstrated the importance of carefully selecting and evaluating machine learning algorithms for specific problems and the need for a thorough analysis and interpretation of the results. The findings and conclusions of this project can inform future research and development in the field of artificial intelligence and machine learning.

**11. APPENDICES**

**11.1 Streamlit Web APP link**

[**https://bank-note-web-app.streamlit.app/**](https://bank-note-web-app.streamlit.app/)

* 1. **GitHub Project link**

[**https://github.com/sryagit/bank\_note\_authentication**](https://github.com/sryagit/bank_note_authentication)

**11.3 Bank Note Web APP Dollar Image**



**Figure 11.1 : Bank Note Web APP Dollar Image**

**12. REFERENCES**

"Intro to Machine Learning | Udacity." Intro to Machine Learning | Udacity. Accessed

April 27, 2016. https://www.udacity.com/course/i

**11.1 Online Resources**

1. [Python](https://www.python.org/doc/), [Numpy](https://numpy.org/doc/stable/), [Pandas](https://pandas.pydata.org/docs/), [Matplotlib](https://matplotlib.org/stable/users/index.html), [Seaborn](https://seaborn.pydata.org/tutorial.html), [Sklearn](https://scikit-learn.org/stable/user_guide.html) & [Streamlit](https://docs.streamlit.io/develop/api-reference/media/st.image)
2. Dataset : <https://archive.ics.uci.edu/dataset/267/banknote+authentication>
3. W3schools : [Python](https://www.w3schools.com/python/default.asp), [Numpy](https://www.w3schools.com/python/numpy/default.asp), [Pandas](https://www.w3schools.com/python/pandas/default.asp), [Matplotlib](https://www.w3schools.com/python/matplotlib_pyplot.asp) & [Seaborn](https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp)
4. Google : [Machine Learning Crash Course](https://developers.google.com/machine-learning/crash-course)
5. W3schools : [Machine Learning](https://www.w3schools.com/ai/ai_machine_learning.asp)
6. GeeksforGeeks : [Machine Learning Algorithms](https://www.geeksforgeeks.org/machine-learning-algorithms/)
7. Great Learning : [Machine-learning-tutorial](https://www.mygreatlearning.com/blog/machine-learning-tutorial/)
8. Javatpoint : [Machine-learning](https://www.javatpoint.com/machine-learning)
9. Datacamp : [learn machine learning](https://www.datacamp.com/category/machine-learning)
10. Simplilearn : [Machine learning Concepts](https://www.simplilearn.com/tutorials/machine-learning-tutorial)
11. Tutorialspoint : [Overall Machine learning](https://www.tutorialspoint.com/machine_learning/index.htm)

**11.2 Books and Authors**

1. Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 4). Springer.
2. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
3. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
4. Norving, Peter, and Stuart Russel. Artificial Intelligence: A Modern Approach. S.l.: Pearson Education Limited, 2013.
5. Quinlan, J. R. (1993). C4. 5: Programs for machine learning. Elsevier.
6. Raschka, S., & Mirjalili, V. (2017). Python Machine Learning. Packt Publishing Ltd.
7. Raschka, S. (2018). Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow. Packt Publishing Ltd.
8. Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach. Pearson Education.
9. Witten, I. H., and Eibe Frank. Data Mining: Practical Machine Learning Tools and Techniques. Amsterdam: Morgan Kaufman, 2005.