

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



TEACHNOOK
592, 3rd Block,
Koramangala, Bengaluru,
Karnataka 560068

Re: Internship Acceptance letter

Dear **Yadav Suraj Ravishankar,**

We are pleased to offer you **Mr. Yadav Suraj Ravishankar**, Student of **BCA Department, SRM UNIVERSITY**, for an **internship** in **DATA SCIENCE** with our **Company Teachnook collaborated with Wissenaire (IIT Bhubaneswar).**

This is an Internship and Training Program. Our goal is for you to learn more about the domain, to get real industrial knowledge & experience.

As we discussed, your internship is expected to last for **2 months** from **February,2024 to March,2024.**

[However, at the sole discretion of the Company, the duration of the internship may be extended or shortened with or without advance notice. During the Internship no leaves will be provided.]

As an intern, you will not be a Company employee. Therefore, you will not receive a salary, wages, or other compensation. In addition, you will not be eligible for any benefits that the Company offers its employees, including, but not limited to, health benefits, holiday pay, vacation pay, sick leave, retirement benefits. You understand that participation in the internship program is not an offer of employment, and successful completion of the internship does not entitle you to employment with the Company.

During your internship, you may have access to confidential, proprietary, and/or trade secret information belonging to the Company. You agree that you will keep all this information strictly.

Confidential and refrain from using it for your own purposes or from disclosing it to anyone outside the Company. In addition, you agree that, upon conclusion of the internship, you will immediately return to the Company all its property, equipment, and documents, including electronically stored information.

TEACHNOOK EDUTECH

No: 592, 3rd Block, Koramangala, Bengaluru, Karnataka 560068
info_hr@teachnook.com Mob: +9163600 93009



By accepting this offer, you agree that you will follow all of the Company's policies that apply to non-employee interns, including the Company's anti-harassment policy.

This letter constitutes the complete understanding between you and the Company regarding your internship and supersedes all prior discussions or agreements. This letter may only be modified by a written agreement signed by both of us.

I hope that your internship with the Company will be successful and rewarding. Please indicate your acceptance of this offer by signing below and returning it to our company desk.

If you have any questions, please do not hesitate to contact us.

Very truly yours,
Saumya Tiwari
Senior HR Manager
TEACHNOOK

I accept Intern with the Company on the terms and conditions set out in this letter.

Date : 04/02/2024

Signature





TEACHNOOK EDUTECH

No: 592, 3rd Block, Koramangala, Bengaluru, Karnataka 560068

info_hr@teachnook.com Mob: +9163600 93009

INTERNSHIP COMPLETION CERTIFICATE




CERTIFICATE OF INTERNSHIP COMPLETION


PRESENTED TO


Suraj Ravishankar Yadav

has successfully completed Mentorship Program on Data Science
from Coratia Technologies, in association with Teachnook. During this internship,
The student has been found to be a keen and enthusiastic candidate.


Certificate ID : TNK2404-01557
Duration : 01/02/2024 - 31/03/2024


Academic Head


SR.HR Manager



Data Science

 **Verified Certificate**

A Verified Certificate from Teachnook can provide a proof for
a student or other institution, an employer or other institution,
that you have successfully completed an online internship.

COURSE COMPLETION CERTIFICATE



Certificate of Course Completion

PRESENTED TO

Suraj Ravishankar Yadav

has successfully completed a course in Data Science
with Teachnook in collaboration with Cognizance'24 IIT Roorkee.
During this course, the student has found to be a keen and enthusiastic candidate.

Certificate ID : TNCC2404-011794

Duration : 01-Feb-2024 To 31-Mar-2024



SENIOR HR MANAGER

ACADEMIC HEAD

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

1.1 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.

1.2 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.

1.3 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.

1.4 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

3.1 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.

3.2 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.

3.3 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.

3.4 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

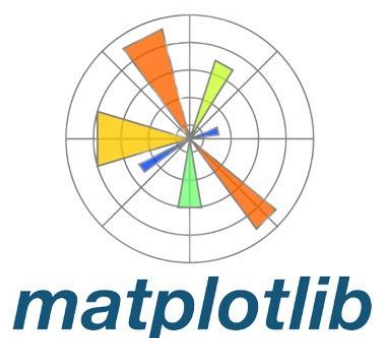


Figure 5.4 : Matplotlib Icon - Plotting Library

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

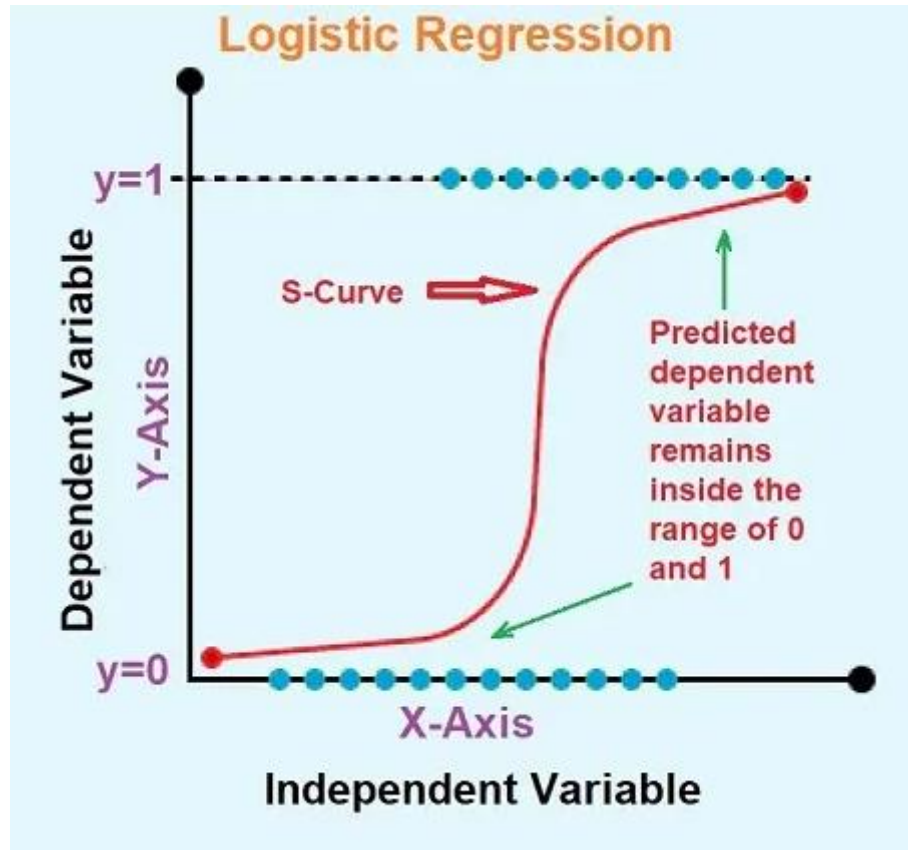


Figure 6.1 : Logistic Regression - Overview

- 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

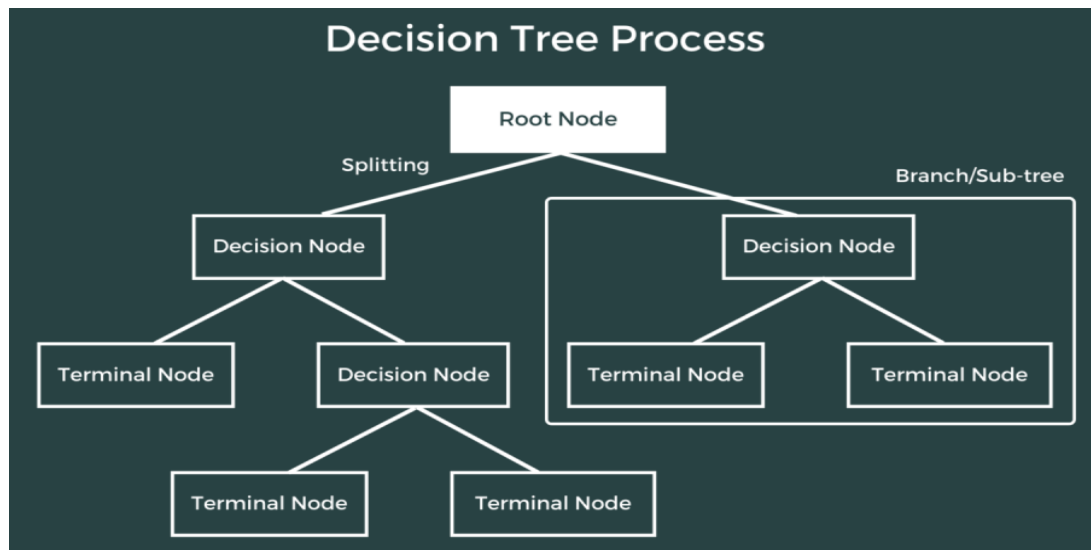


Figure 6.2 : Decision Tree Classifier - Overview

- 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

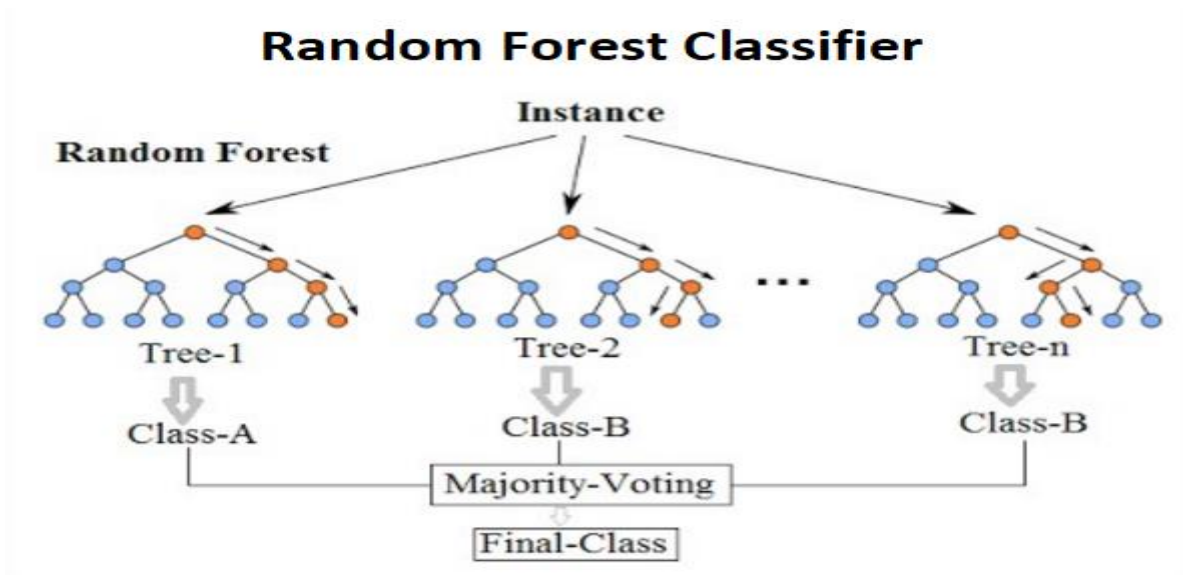


Figure 6.3 : Random Forest Classifier - Overview

- 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)

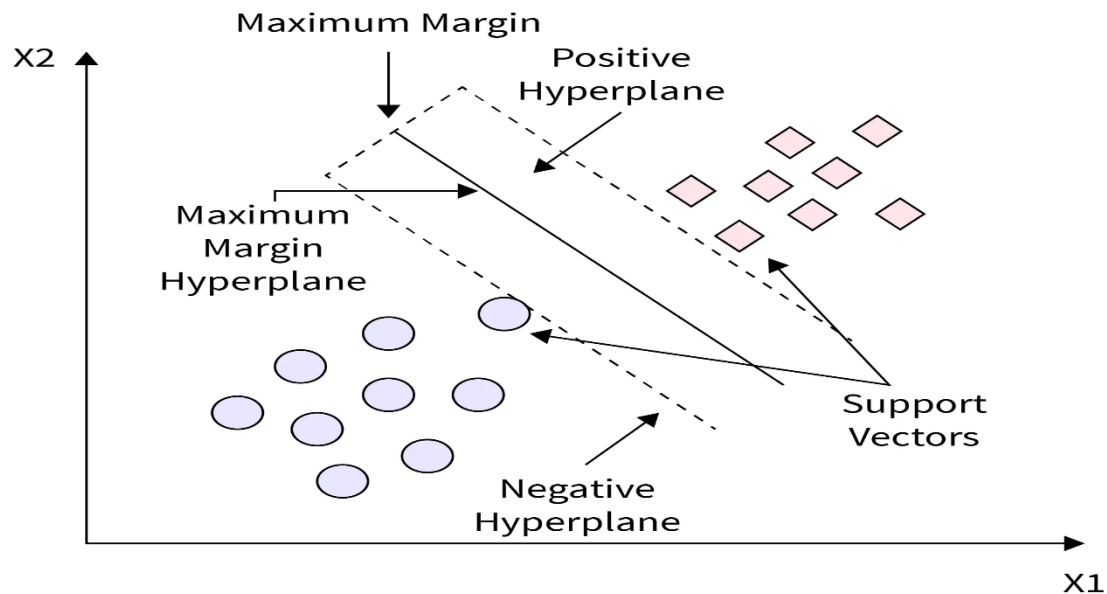


Figure 6.4 : Support Vector Machine (SVM) - Overview

- 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

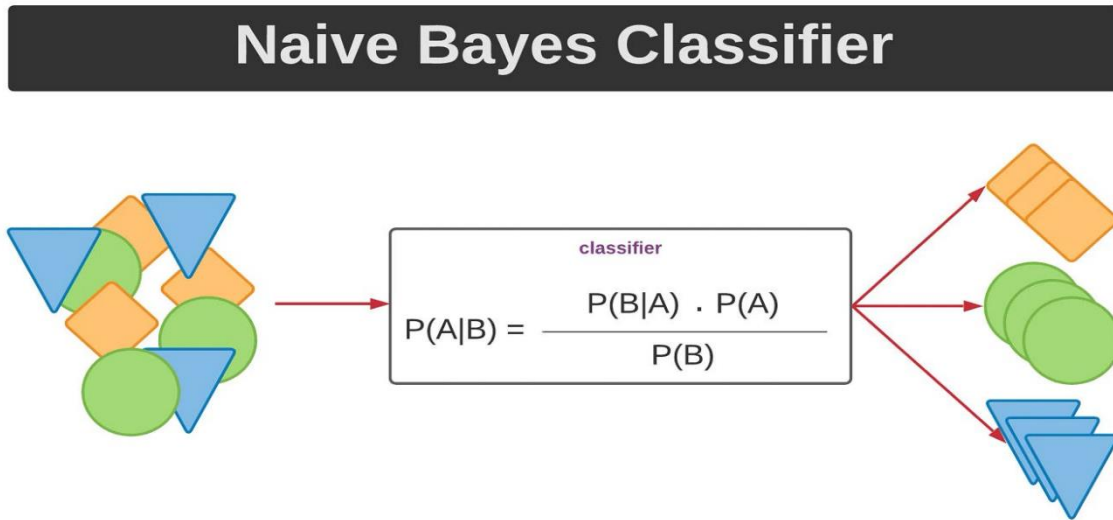


Figure 6.5 : Naive Bayes Classifier - Overview

- 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

7.1 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')
```

7.2 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()
```

7.3 Requirements.txt file

numpy==1.26.4

pandas==2.2.2

matplotlib==3.9.0

seaborn==0.13.2

scikit-learn==1.4.2

jjoblib==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

```
# Importing required libraries and modules
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import joblib

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
```

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

```
# Check the versions of Libraries
print('Python: {}'.format(sys.version))
print('numpy: {}'.format(numpy.__version__))
print('pandas: {}'.format(pandas.__version__))
print('matplotlib: {}'.format(matplotlib.__version__))
print('seaborn: {}'.format(seaborn.__version__))
print('sklearn: {}'.format(sklearn.__version__))
```

Python: 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
numpy: 1.26.4
pandas: 2.2.1
matplotlib: 3.8.4
seaborn: 0.12.2
sklearn: 1.4.2

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

```
# Suppress display of warnings
import warnings
warnings.filterwarnings('ignore')
```

Figure 8.3 : Suppressing 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

```
# Loading the dataset
df = pd.read_csv('bank_note_authentication.csv')
df
```

	variance	skewness	curtosis	entropy	class
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

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

```
# Checking the dimensions of dataset  
df.shape  
  
(1372, 5)
```

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

```
# Checking columns of dataset  
df.columns  
  
Index(['variance', 'skewness', 'curtosis', 'entropy', 'class'], dtype='object')
```

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

```
# Checking for missing or null values
df.isnull().sum()

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

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

```
# Printing summary of DataFrame
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype  
---  -
0   variance    1372 non-null   float64
1   skewness    1372 non-null   float64
2   curtosis    1372 non-null   float64
3   entropy     1372 non-null   float64
4   class       1372 non-null   int64   
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

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

```
# Generating descriptive statistics
df.describe()
```

	variance	skewness	curtosis	entropy	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

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

```
# Computing pairwise correlations
df.corr()
```

	variance	skewness	curtosis	entropy	class
variance	1.000000	0.264026	-0.380850	0.276817	-0.724843
skewness	0.264026	1.000000	-0.786895	-0.526321	-0.444688
curtosis	-0.380850	-0.786895	1.000000	0.318841	0.155883
entropy	0.276817	-0.526321	0.318841	1.000000	-0.023424
class	-0.724843	-0.444688	0.155883	-0.023424	1.000000

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

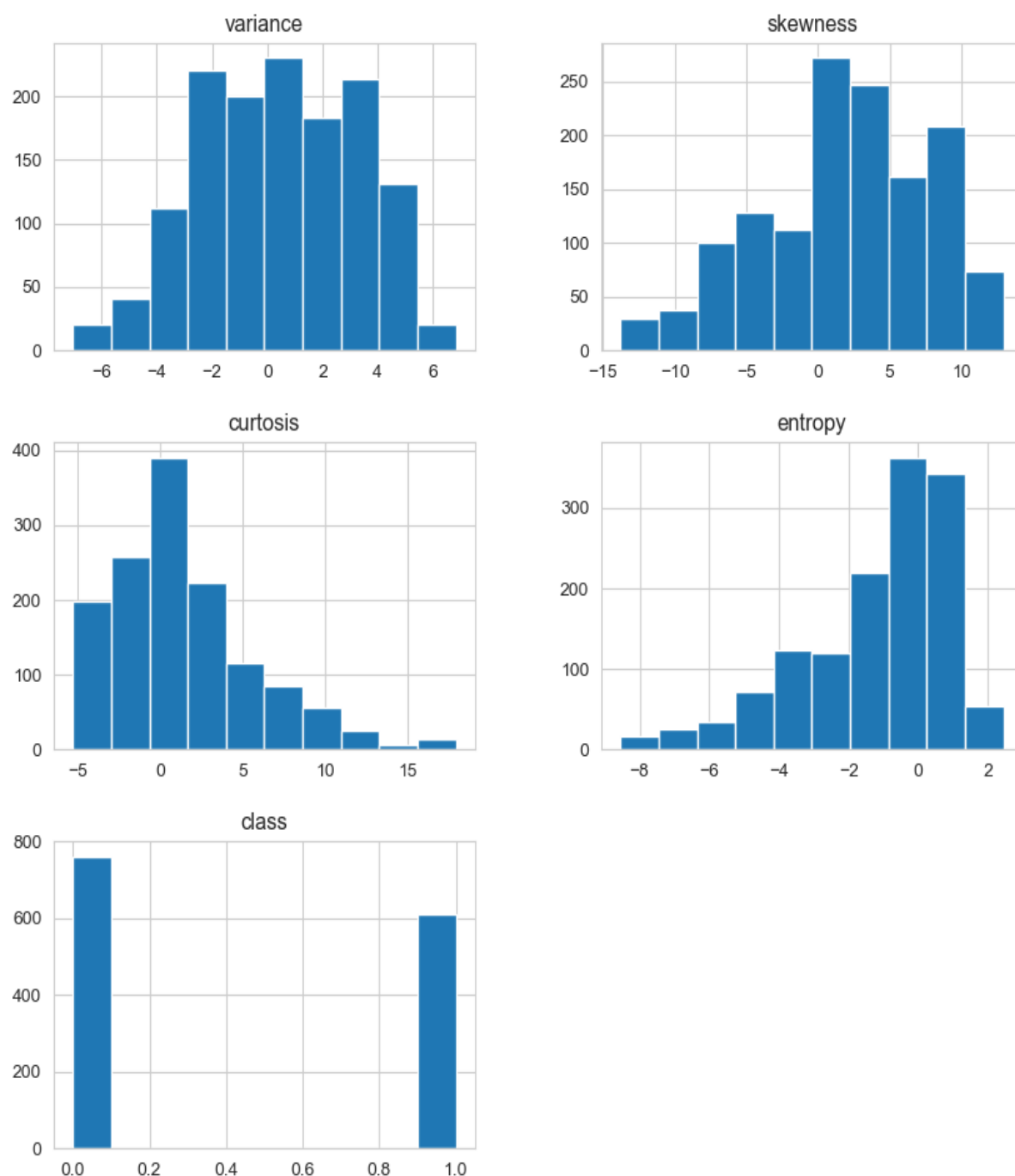


Figure 8.11 : Distribution of Variables

Conclusion:

- 1) The variance, skewness, and kurtosis 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.

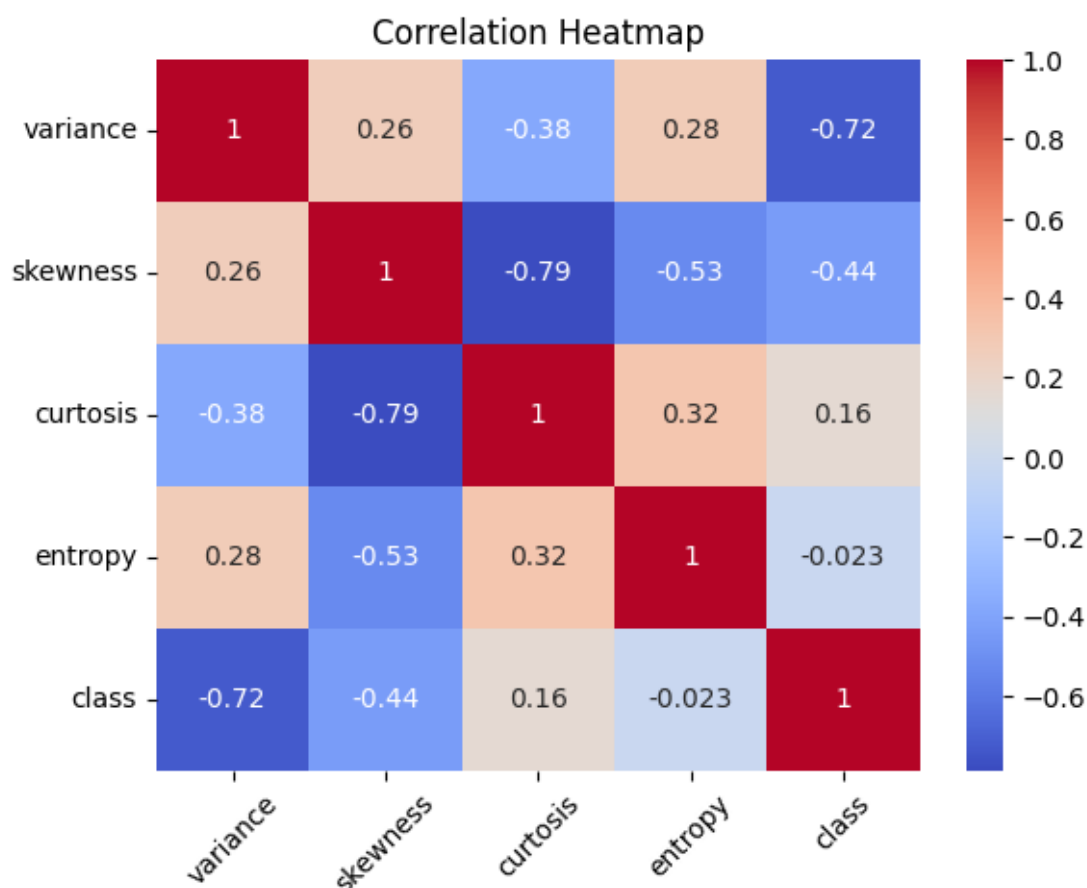


Figure 8.12 : Correlation Heatmap

➤ **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 there is a multicollinearity 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.

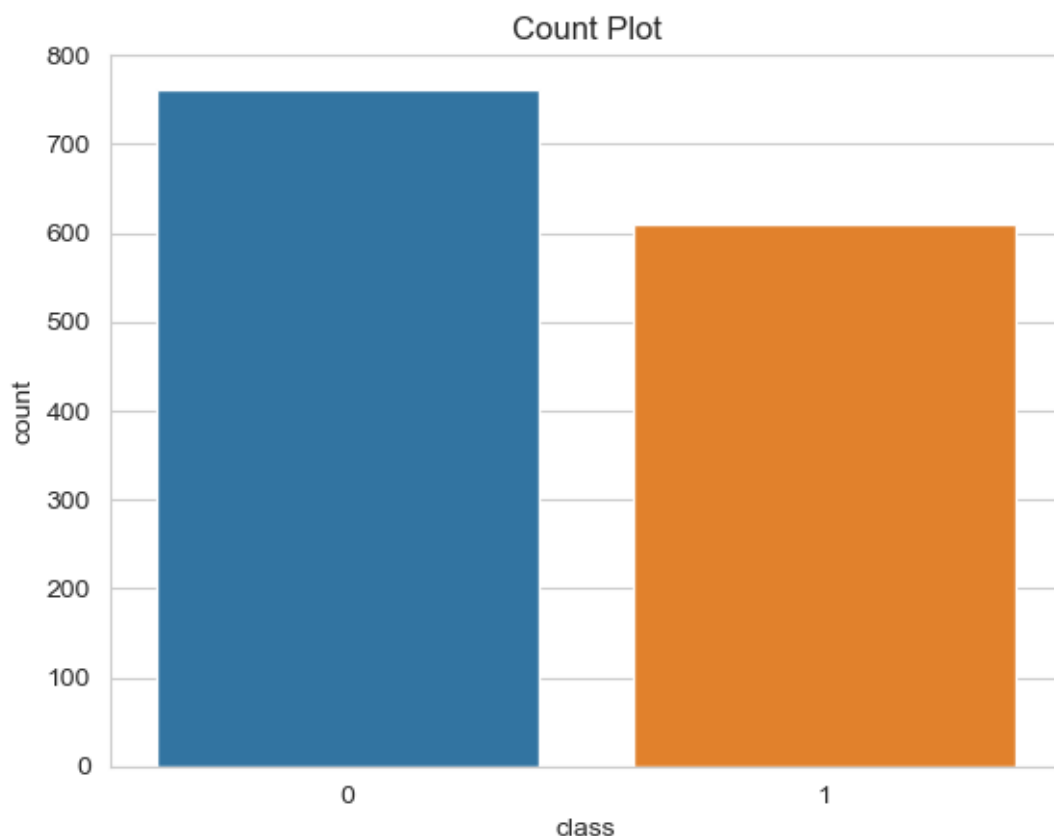


Figure 8.13 : Label Data Countplot

- **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.

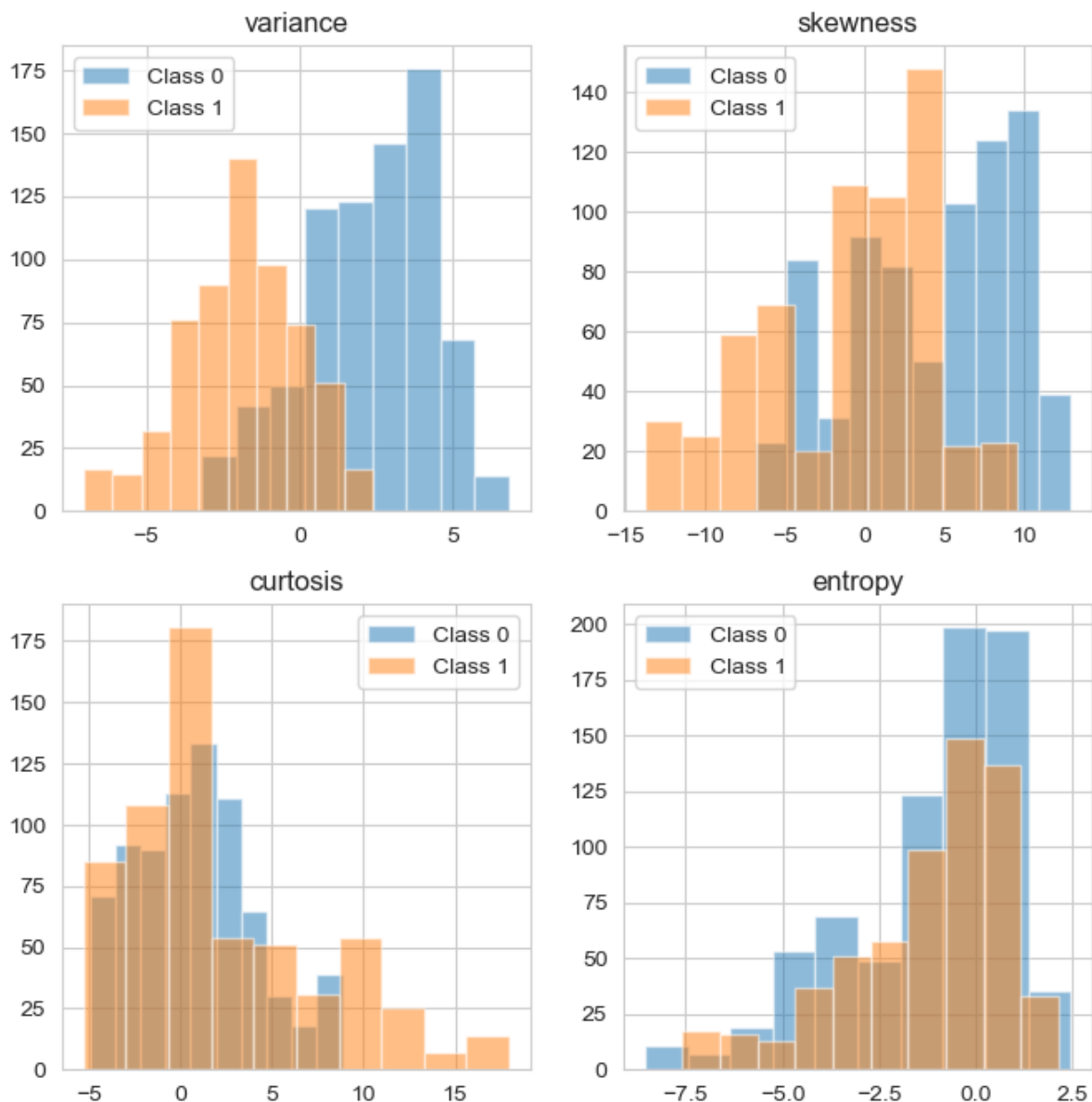


Figure 8.14 : Class Distribution of Features

➤ **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.

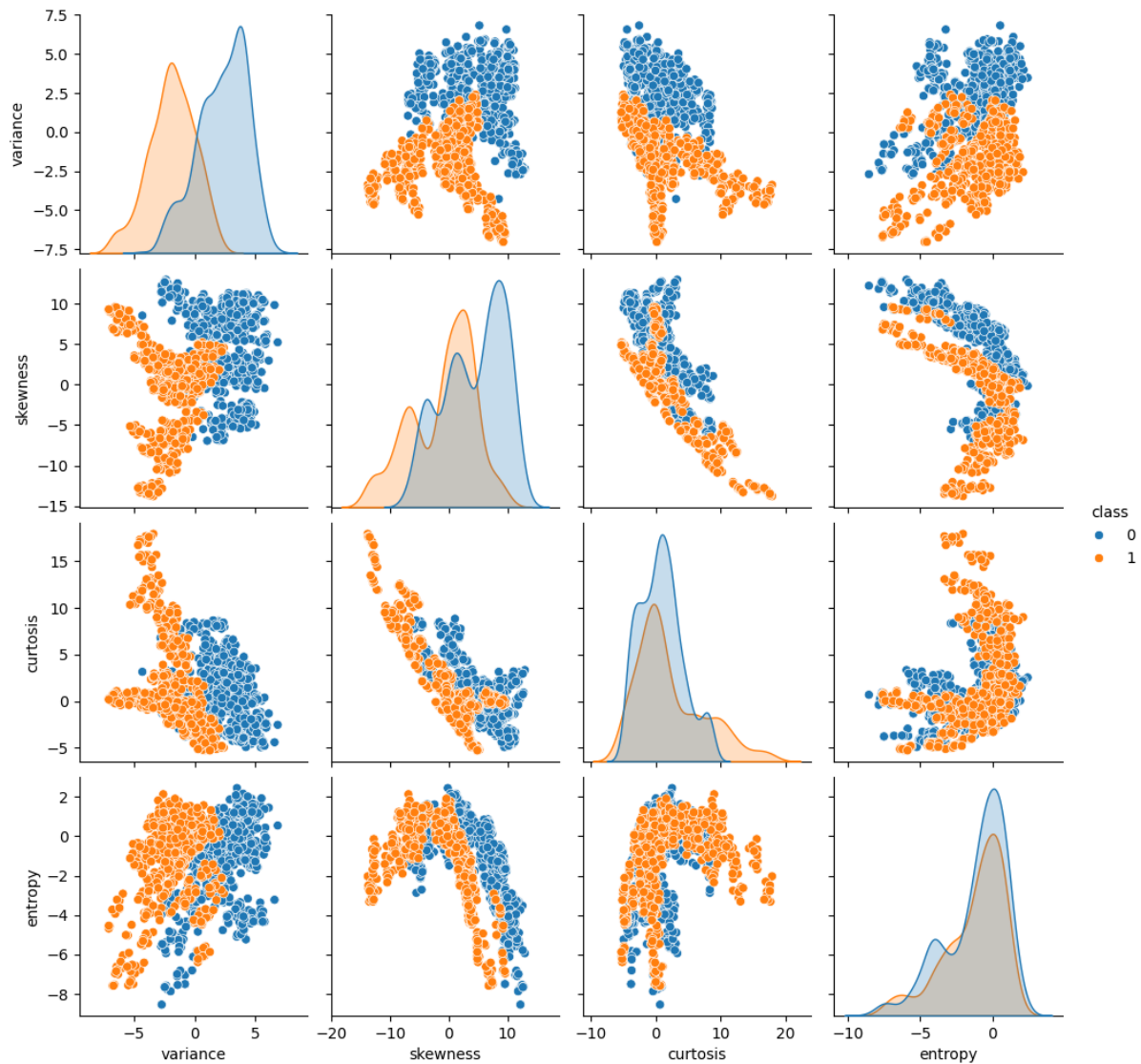


Figure 8.15 : Features Pairplot

➤ **Conclusions:**

- There are clear separations shown, especially for pairs of features having 'variance'.
- The kurtosis-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.

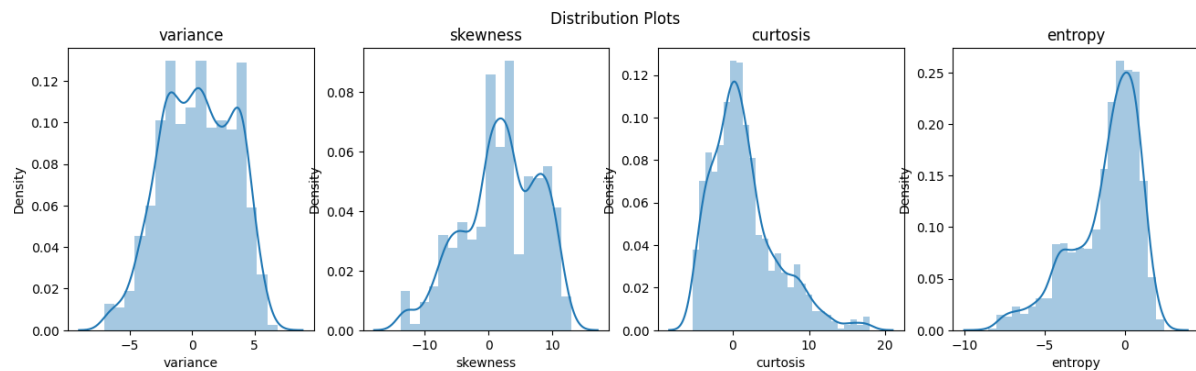


Figure 8.16 : Distribution Plots

➤ **Conclusions:**

- 1) There may be outliers in entropy and kurtosis column.
- 2) Data is not normalized.
- 3) Since we gonna use trees in our model above things don't effect model much.

8.8 Train-Test Split

8.8.1 Splitting the Dataset into Features and Target Variable

```
# Split the dataset into features and target variable
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X.head(), y.head()
```

```
(   variance  skewness  kurtosis  entropy
0    3.62160    8.6661   -2.8073 -0.44699
1    4.54590    8.1674   -2.4586 -1.46210
2    3.86600   -2.6383    1.9242  0.10645
3    3.45660    9.5228   -4.0112 -3.59440
4    0.32924   -4.4552    4.5718 -0.98880,
0      0
1      0
2      0
3      0
4      0
Name: class, dtype: int64)
```

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

```
# 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)
X_train.head(2), X_test.head(2), y_train.head(2), y_test.head(2)
```

```
(      variance  skewness  curtosis  entropy
529   -1.3885   12.5026    0.69118 -7.548700
243    2.7744    6.8576   -1.06710  0.075416,
      variance  skewness  curtosis  entropy
430    1.56910    6.3465   -0.1828  -2.4099
588   -0.27802    8.1881   -3.1338  -2.5276,
529      0
243      0
Name: class, dtype: int64,
430      0
588      0
Name: class, dtype: int64)
```

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

```
# 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)

Shape of X_train: (1097, 4)
Shape of y_train: (1097,)
Shape of X_test: (275, 4)
Shape of y_test: (275,)
```

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

```
# Apply StandardScaler to the training data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_train_scaled

array([[ -0.6391558 ,  1.80557961, -0.18836535, -3.05096841],
       [ 0.82188925,  0.85239902, -0.59407847,  0.60345479],
       [-1.65703344, -1.63328321,  2.38386151, -0.34235536],
       ...,
       [-2.62138845,  1.26364283, -0.3095615 , -1.608634  ],
       [-1.36636167,  0.14870015, -0.31055139,  0.07724503],
       [-1.33045764, -1.52810408,  2.29230217,  0.29354966]])
```

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

```
# Use the same scaler to transform the test data
X_test_scaled = scaler.transform(X_test)
X_test_scaled

array([[ 0.39886742,  0.76609776, -0.39003127, -0.58781728],
       [-0.24941276,  1.07705921, -1.07095774, -0.64423373],
       [-0.13359364,  0.88524111, -0.82182311, -0.94294857],
       ...,
       [ 1.06342549, -1.05676514,  0.46960667,  0.64126101],
       [-0.24940574,  1.07705921, -1.07095774, -0.64423373],
       [ 0.82494267,  1.54841503, -1.16860874, -1.5435897 ]])
```

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

```
# 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)
```

Figure 8.22 : Creating Logistic Regression Classifier

8.10.2 Creating Decision Tree Classifier

```
# 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)
```

Figure 8.23 : Creating Decision Tree Classifier

8.10.3 Creating Random Forest Classifier

```
# 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)
```

Figure 8.24 : Creating Random Forest Classifier

8.10.4 Creating Support Vector Classifier

```
# 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)
```

Figure 8.25 : Creating Support Vector Classifier

8.10.5 Creating Naive Bayes Classifier

```
# 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)
```

Figure 8.26 : Creating Naive Bayes Classifier

8.11 Model Evaluation

8.11.1 LR Accuracy Score and Confusion Matrix

```
lr_accuracy = accuracy_score(y_test, lr_y_pred)*100
print("Logistic Regression Accuracy Score:", lr_accuracy)
```

```
Logistic Regression Accuracy Score: 97.81818181818181
```

Figure 8.27 : Logistic Regression Accuracy Score

```
lr_cm = confusion_matrix(y_test, lr_y_pred)
print("Logistic Regression Confusion Matrix:\n\n", lr_cm)
```

```
Logistic Regression Confusion Matrix:
```

```
[[144  4]
 [ 2 125]]
```

Figure 8.28 : Logistic Regression Confusion Matrix

8.11.2 Logistic Regression Classification Report

```
lr_cr = classification_report(y_test, lr_y_pred)
print("Logistic Regression Classification Report:\n\n", lr_cr)
```

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	148
1	0.97	0.98	0.98	127
accuracy			0.98	275
macro avg	0.98	0.98	0.98	275
weighted avg	0.98	0.98	0.98	275

Figure 8.29 : Logistic Regression Classification Report

8.11.3 DTC Accuracy Score and Confusion Matrix

```
dt_accuracy = accuracy_score(y_test, dt_y_pred)*100
print("Decision Tree Classifier Accuracy Score:", dt_accuracy)
```

Decision Tree Classifier Accuracy Score: 97.0909090909091

Figure 8.30 : Decision Tree Classifier Accuracy Score

```
dt_cm = confusion_matrix(y_test, dt_y_pred)
print("Decision Tree Classifier Confusion Matrix:\n\n", dt_cm)
```

Decision Tree Classifier Confusion Matrix:

```
[[147  1]
 [ 7 120]]
```

Figure 8.31 : Decision Tree Classifier Confusion Matrix

8.11.4 Decision Tree Classifier Classification Report

```
dt_cr = classification_report(y_test, dt_y_pred)
print("Decision Tree Classifier Classification Report:\n\n", dt_cr)
```

Decision Tree Classifier Classification Report:

	precision	recall	f1-score	support
0	0.95	0.99	0.97	148
1	0.99	0.94	0.97	127
accuracy			0.97	275
macro avg	0.97	0.97	0.97	275
weighted avg	0.97	0.97	0.97	275

Figure 8.32 : Decision Tree Classifier Classification Report

8.11.5 Random Accuracy Score and Confusion Matrix

```
rf_accuracy = accuracy_score(y_test, rf_y_pred)*100
print("Random Forest Classifier Accuracy Score:", rf_accuracy)
```

Random Forest Classifier Accuracy Score: 99.27272727272727

Figure 8.33 : Random Forest Classifier Accuracy Score

```
rf_cm = confusion_matrix(y_test, rf_y_pred)
print("Random Forest Classifier Confusion Matrix:\n\n", rf_cm)
```

Random Forest Classifier Confusion Matrix:

```
[[148  0]
 [ 2 125]]
```

Figure 8.34 : Random Forest Classifier Confusion Matrix

8.11.6 Random Forest Classifier Classification Report

```
rf_cr = classification_report(y_test, rf_y_pred)
print("Random Forest Classifier Classification Report:\n\n", rf_cr)
```

Random Forest Classifier Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	148
1	1.00	0.98	0.99	127
accuracy			0.99	275
macro avg	0.99	0.99	0.99	275
weighted avg	0.99	0.99	0.99	275

Figure 8.35 : Random Forest Classifier Classification Report

8.11.7 Support Accuracy Score and Confusion Matrix

```
svm_accuracy = accuracy_score(y_test, svm_y_pred)*100
print("Support Vector Classifier Accuracy Score:", svm_accuracy)
```

Support Vector Classifier Accuracy Score: 98.54545454545455

Figure 8.36 : Support Vector Classifier Accuracy Score

```
svm_cm = confusion_matrix(y_test, svm_y_pred)
print("Support Vector Classifier Confusion Matrix:\n\n", svm_cm)
```

Support Vector Classifier Confusion Matrix:

```
[[146  2]
 [ 2 125]]
```

Figure 8.37 : Support Vector Classifier Confusion Matrix

8.11.8 Support Vector Classifier Classification Report

```
svm_cr = classification_report(y_test, svm_y_pred)
print("Support Vector Classifier Classification Report:\n\n", svm_cr)
```

Support Vector Classifier Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	148
1	0.98	0.98	0.98	127
accuracy			0.99	275
macro avg	0.99	0.99	0.99	275
weighted avg	0.99	0.99	0.99	275

Figure 8.38 : Support Vector Classifier Classification Report

8.11.9 Naive Accuracy Score and Confusion Matrix

```
nb_accuracy = accuracy_score(y_test, nb_y_pred)*100
print("Naive Bayes Classifier Accuracy Score:", nb_accuracy)
```

Naive Bayes Classifier Accuracy Score: 80.72727272727272

Figure 8.39 : Naive Bayes Classifier Accuracy Score

```
nb_cm = confusion_matrix(y_test, nb_y_pred)
print("Naive Bayes Classifier Confusion Matrix:\n\n", nb_cm)
```

Naive Bayes Classifier Confusion Matrix:

```
[[133  15]
 [ 38  89]]
```

Figure 8.40 : Naive Bayes Classifier Confusion Matrix

8.11.10 Naive Bayes Classifier Classification Report

```
nb_cr = classification_report(y_test, nb_y_pred)
print("Naive Bayes Classifier Classification Report:\n", nb_cr)
```

```
Naive Bayes Classifier Classification Report:
              precision    recall  f1-score   support

     0       0.78         0.90         0.83         148
     1       0.86         0.70         0.77         127

 accuracy          0.81         0.81         0.81         275
 macro avg         0.82         0.80         0.80         275
 weighted avg         0.81         0.81         0.80         275
```

Figure 8.41 : Naive Bayes Classifier Classification Report

8.12 Summarizing Accuracy Scores

```
# 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)
```

```
Logistic Regression Accuracy: 97.81818181818181
Decision Tree Classifier Accuracy: 97.0909090909091
Random Forest Classifier Accuracy: 99.27272727272727
Support Vector Classifier Accuracy: 98.54545454545455
Naive Bayes Classifier Accuracy: 80.72727272727272
```

Figure 8.42 : Summarizing Accuracy Scores

8.13 Visualizing Accuracy Scores

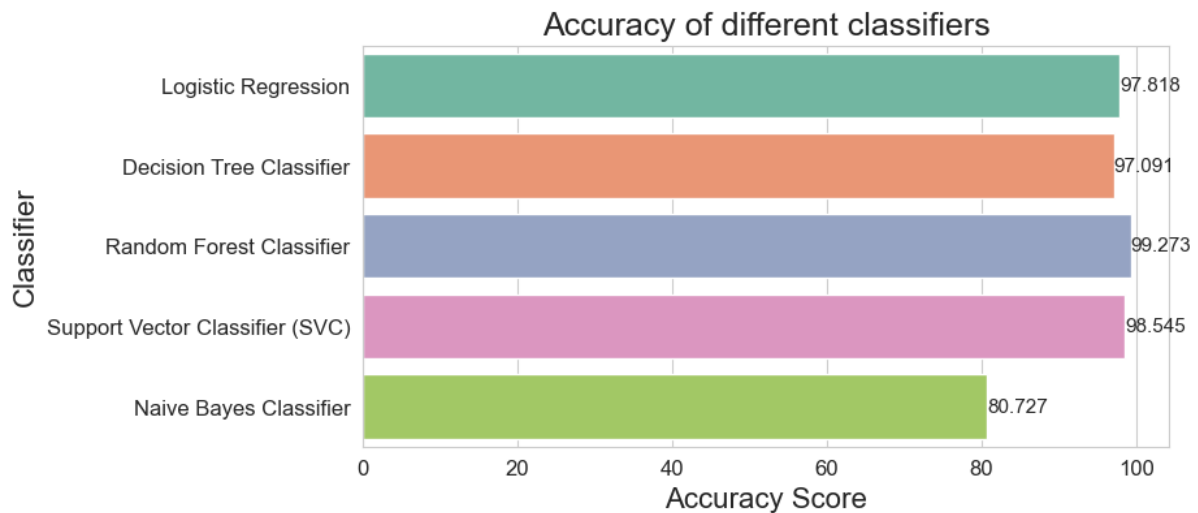


Figure 8.43 : Visualizing Accuracy Scores

- 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

```
## 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

[0]
[1]
[0]
[1]
```

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

```
# Create a joblib file using serialization
import joblib
joblib.dump(rf_clf, 'classifier.joblib')

['classifier.joblib']
```

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

```
# Loading the saved model  
model = joblib.load('classifier.joblib')
```


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

REAL or FAKE ?



Bank Note Authentication Web APP

Variance

Skewness


Curtosis

Entropy

Figure 9.1 : Web APP Interface

9.2 Prediction with Random Sample Data 1

REAL or FAKE ?



Bank Note Authentication Web APP

Variance

3.45660

Skewness

9.52280

Curtosis

-4.0112

Entropy

-3.59440

Get Prediction

Result : 0

0 = banknote is forged

1 = banknote is genuine

About

Classifier : Random Forest


Accuracy : 99.27 %

Built by : Suraj R. Yadav

Figure 9.2 : Prediction with Random Sample Data 1

9.3 Prediction with Random Sample Data 2

REAL or FAKE ?



Bank Note Authentication Web APP

Variance

-3.56370

Skewness

-8.38270

Curtosis

12.3930

Entropy

-1.28230

Get Prediction

Result : 1

0 = banknote is forged

1 = banknote is genuine

About

Classifier : Random Forest


Accuracy : 99.27 %

Built by : Suraj R. Yadav

Figure 9.3 : Prediction with Random Sample Data 2

9.4 Prediction with Random Sample Data 3

REAL or FAKE ?



Bank Note Authentication Web APP

Variance

4.54590

Skewness

8.16740

Curtosis

-2.4586

Entropy

-1.46210

Get Prediction

Result : 0

0 = banknote is forged

1 = banknote is genuine

About

Classifier : Random Forest


Accuracy : 99.27 %

Built by : Suraj R. Yadav

Figure 9.4 : Prediction with Random Sample Data 3

9.5 Prediction with Random Sample Data 4

REAL or FAKE ?



Bank Note Authentication Web APP

Variance

-1.38870

Skewness

-4.87730

Curtosis

6.4774

Entropy

0.34179

Get Prediction

Result : 1

0 = banknote is forged

1 = banknote is genuine

About

Classifier : Random Forest

Accuracy : 99.27 %

Built by : Suraj R. Yadav

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/>

11.2 GitHub Project link

https://github.com/sryagit/bank_note_authentication

11.3 Bank Note Web APP Dollar Image

REAL or FAKE ?



Figure 11.1 : Bank Note Web APP Dollar Image

12. REFERENCES

11.1 Online Resources

- [1] [Python](#), [Numpy](#), [Pandas](#), [Matplotlib](#), [Seaborn](#), [Sklearn](#) & [Streamlit](#)
- [2] Dataset : <https://archive.ics.uci.edu/dataset/267/banknote+authentication>
- [3] W3schools : [Python](#), [Numpy](#), [Pandas](#), [Matplotlib](#) & [Seaborn](#)
- [4] Google : [Machine Learning Crash Course](#)
- [5] W3schools : [Machine Learning](#)
- [6] GeeksforGeeks : [Machine Learning Algorithms](#)
- [7] Great Learning : [Machine-learning-tutorial](#)
- [8] Javatpoint : [Machine-learning](#)
- [9] Datacamp : [learn machine learning](#)
- [10] Simplilearn : [Machine learning Concepts](#)
- [11] Tutorialspoint : [Overall Machine learning](#)

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.I.: 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.