Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Banknotes Authentication Using Machine Learning Models

Parallel and Distributed Computing
CSE4001

Submitted by

Garv Mittal – 20BCE2857

Aryan Agrawal – 20BCE2876

Somya Rathi – 20BCE2323

Rohit Barik – 20BCE2874

Aryan Sharma – 20BCE0952

Abstract

In cash transactions, the biggest challenge faced is counterfeit or fake notes.This issue is only expanding due to the enhancement in the technology.Therefore, this issue of differentiating between a counterfeit or fake banknote and the actual note using an automatic machine has become more and more crucial. Counterfeit or fake notes are an issue of concern to almost every country and it has become a very acute problem.

Introduction

Banknote authentication is a critical process in ensuring the security of financial transactions. Counterfeit banknotes can cause significant damage to the economy and the trust of consumers in financial institutions. In recent years, there has been a growing interest in using machine learning models to authenticate banknotes.

The aim of this project is to develop a banknote authentication system using machine learning models. The system will be trained on a dataset of images of genuine and counterfeit banknotes. The dataset will be preprocessed and features extracted before being used to train the machine learning models. The project will explore various machine learning algorithms, including support vector machines (SVMs), random forests, and logistic regression .

To counter this problem of fake notes, there is a need to design such an authentication system that can differentiate between the fake notes and the real one. This project describes an approach for checking banknotes.

In this project,four columns of features which are extracted are used and we apply our ML algorithm on it.

METHODOLOGY:

In this proposed project four columns of features which are extracted are used and we apply our ML algorithm on it. The approach consists of a number of characteristics includingVariance, Skewness, Curtosis, Entropy. These characteristics will be used in the authentication algorithm to train our model on real notes and after training the model we will check and test the different models on some of the real and fake notes to determine the feasibility of the different models.

In the process of building a machine learning model, exploratory analysis is often used to gain a better understanding of the data and its various attributes. Exploratory analysis involves techniques such as data visualization, descriptive statistics, and feature engineering, among others, to explore and understand the relationships between different variables in the dataset.

Once a thorough exploration of the data has been completed, the next step is to build a machine learning model that can accurately predict the outcome or target variable based on the features or input variables. This can involve various machine learning algorithms such as linear regression, decision trees, or neural networks, depending on the nature of the data and the specific problem at hand.

The ultimate goal of building a machine learning model is to produce accurate and reliable predictions that can be used to inform decision-making or gain insights into complex phenomena. By using exploratory analysis to understand the attributes of the data, we can build a more robust and effective machine learning model that is better suited to the task at hand.

Literature Survey

Serial Number	Title	Author	Year of Publication	Brief Summary
1	Big data machine learning using Apache Spark MLlib	Mehdi Assefi, Ehsun Behravesh, Guangchi Liu, Ahmed P. Tafti	2017	The paper explores the use of Apache Spark MLlib as an open source, distributed, scalable, and platform independent machine learning library for big data analytics. It provides functionalities for regression, classification, dimension reduction, clustering, and rule extraction.
2	Higgs Boson Discovery using Machine Learning Methods with Pyspark	Mourad Azharia, Abdallah Abardab, Badia Ettakia, Jamal Zerouaouia, Mohamed Dakkond	2020	The paper proposes the use of four machine learning methods (Logistic Regression, Decision Tree, Random Forest, and Gradient Boosted Tree) to solve the Higgs Boson Classification Problem using the Pyspark environment. They compare the accuracy and AUC metrics of those ML methods using large datasets collected from public sites.
3	Call Churn Prediction with PySpark	Mark Sheridan Nonghuloo, Rangapuram Aravind Reddy, Ganji Manideep, M. R. SarathVamsi, K. Lavanya	2020	The paper focuses on customer retention by predicting which customers are likely to cancel a subscription to a service based on their call data records. The authors perform churn prediction models with PySpark using customer data analysis. PySpark processes huge datasets at minimal time and handles synchronization points and errors easily.
4	Authentication of Bank Notes and Image Classification Through Artificial and Convolutional Neural Network	MD. Touhidul Islam	2018	The paper uses two common applications of image processing with the help of neural network: detection of counterfeit bank notes using Artificial Neural Network and unstructured image classification problem through Convolutional Neural Network. The structured image dataset of bank notes provided by the UCI Machine Learning Repository and the famous Kaggle dataset were used for these applications.

Serial Number	Title	Author	Year of Publication	Brief Summary
5	Bank Note Authentication: A Genetic Algorithm Supported Neural based Approach	Spandan Sen Sarma	2016	The paper proposes the use of a Neural Network trained with Genetic Algorithm to accurately separate original bank notes from forged ones. The proposed method's results were compared to a well-known Multilayer Perceptron Feed-Forward Network and another Neural Network.
6	Employing Multiple-Kernel Support Vector Machines for Counterfeit Banknote Recognition	Chi-YuanYeh, Wen-PinSu, Shie-JueLee	2011	The paper proposes a system based on multiple-kernel support vector machines for counterfeit banknote recognition. Each banknote is divided into partitions, and the luminance histograms of the partitions are taken as input. Each partition is associated with its own kernels, and linear regression is applied to obtain the final result.

Novelty

Detection is a technique used to identify anomalies or unusual patterns in data that may indicate the presence of counterfeit banknotes. To achieve this, several machine learning models can be utilized, such as Random Forest Algorithm, Logistic regression, Decision Tree Model, Support Vector Machine, and Gradient Boosting. After evaluating the performance of these models on a given dataset, it was observed that Support Vector Machine (SVM) provided the highest accuracy in identifying fake notes. Hence, SVM can be considered as an effective method to detect counterfeit notes and prevent financial losses caused by fraud.

In the context of detecting fake banknotes, novelty refers to the identification of unusual or anomalous patterns in the data that can indicate the presence of counterfeit notes. The novelty detection approach is based on the assumption that counterfeit notes have different characteristics or features compared to genuine banknotes. Therefore, by training machine learning models on a dataset of genuine banknotes, the models can learn to recognize these distinguishing features and identify counterfeit notes as anomalies or novelties in the data. In this way, novelty detection can be a powerful tool in preventing financial losses due to fraud.

MACHINE LEARNING MODELS USED

1. Random Forest Algorithm:

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks). Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

One of the biggest advantages of random forest is its versatility. It can be used for both regression and classification tasks, and it’s also easy to view the relative importance it assigns to the input features.

Random forest is also a very handy algorithm because the default hyperparameters it uses often produce a good prediction result. Understanding the hyperparameters is pretty straightforward, and there's also not that many of them.

And, of course, random forest is a predictive modeling tool and not a descriptive tool, meaning if you're looking for a description of the relationships in your data, other approaches would be better.

2. Logistic regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. It is a type of generalized linear model that is used for predicting binary outcomes (e.g., yes/no, success/failure) or nominal outcomes (e.g., categories such as red/green/blue).

The logistic regression model is a type of regression analysis that estimates the probability of the occurrence of an event by fitting data to a logistic function. The logistic function is an S-shaped curve that maps any real-valued input to a value between 0 and 1. The logistic function is used to model the probability of a binary outcome as a function of one or more predictors (e.g., age, gender, income).

In logistic regression, the dependent variable is binary or categorical and represented by 0s and 1s. The independent variables can be continuous or categorical. The logistic regression model estimates the probability that the dependent variable equals 1 given the values of the independent variables. The model uses a set of coefficients to estimate the impact of each independent variable on the dependent variable.

Logistic regression is a popular method for predicting outcomes in a wide range of fields, including medicine, social sciences, and business. It is widely used in machine learning and data science for classification problems.

3. Decision Tree Model

A decision tree model is a type of supervised learning algorithm that is commonly used for classification and regression problems. It is a tree-like model where each node represents a feature or attribute, each branch represents a decision rule, and each leaf node represents a class label or a numerical value.

In a decision tree, the algorithm iteratively partitions the data into subsets based on the values of the features, creating decision rules that split the data into smaller and smaller subsets until the leaf nodes are reached, which represent the final predictions. The decision rules are determined by finding the features that provide the most information gain or minimize the impurity in the target variable.

Decision trees are easy to interpret and visualize, making them useful for understanding how a model is making decisions. They can handle both categorical and numerical data, and they can also handle missing data. However, decision trees can suffer from overfitting, where the model becomes too complex and fits the noise in the data, which can lead to poor generalization performance.

There are several variations of decision trees, including Random Forests and Gradient Boosted Trees, which aim to improve the performance and robustness of the model. Decision tree models are widely used in various fields, including finance, healthcare, and marketing, as well as in machine learning competitions and data science projects.

4. Support Vector Machine

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for both classification and regression tasks.

SVMs are particularly effective in cases where there is a clear margin of separation between classes, which makes them popular in image classification, text classification, and other data analysis tasks.

The basic idea behind SVMs is to find the hyperplane that best separates the data into classes. In a two-dimensional space, a hyperplane is simply a line that separates the data into two classes. In higher dimensions, a hyperplane is a surface that separates the data into multiple classes. The goal is to find the hyperplane that maximizes the margin between the classes, where the margin is defined as the distance between the hyperplane and the nearest data points of each class.

SVMs can handle both linearly separable and non-linearly separable data by transforming the data into a higher-dimensional space, where the data can be separated by a hyperplane. This is achieved by using a kernel function that maps the data into a higher-dimensional space.

SVMs have several advantages over other classification algorithms, including the ability to handle high-dimensional data, the ability to handle both linearly separable and non-linearly separable data, and the ability to avoid overfitting. SVMs are widely used in various fields, including bioinformatics, finance, and image analysis, as well as in machine learning competitions and data science projects.

5. Gradient Boosting

Gradient boosting is a machine learning technique that is used for both regression and classification problems. It is an ensemble learning method that combines several weak learning models (usually decision trees) to create a stronger predictive model. Gradient boosting models are particularly effective in cases where there is a large amount of data, and the individual models have high variance.

The basic idea behind gradient boosting is to sequentially add models to the ensemble, where each new model is trained to correct the errors of the previous models. The models are added to the ensemble using a gradient descent algorithm, where the objective is to minimize the loss function of the model.

In the case of decision trees, each new tree is trained to fit the residual errors of the previous trees. The residual errors are the differences between the predicted values and the true values of the target variable. The final prediction of the ensemble is the sum of the predictions of all the models.

Gradient boosting has several advantages over other machine learning techniques, including the ability to handle both categorical and numerical data, the ability to handle missing data, and the ability to handle high-dimensional data. Gradient boosting models are widely used in various fields, including finance, marketing, and natural language processing, as well as in machine learning competitions and data science projects. Some popular implementations of gradient boosting include XGBoost, LightGBM, and CatBoost.

TECHNOLOGIES USED

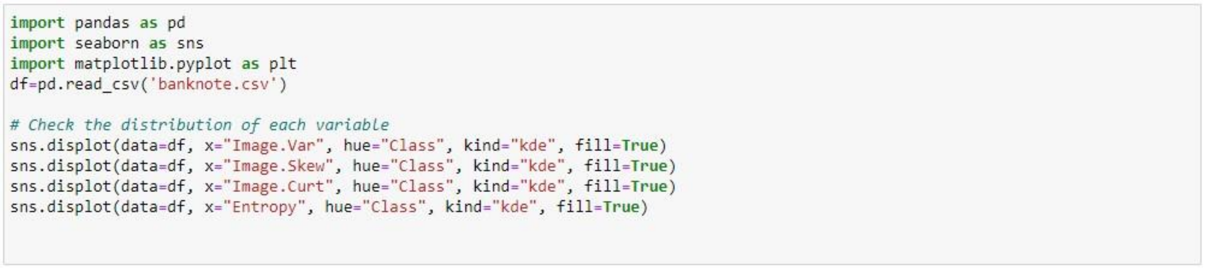
Language - Python 3

Environment - Jupyter notebook

Various ML models

EXPERIMENT

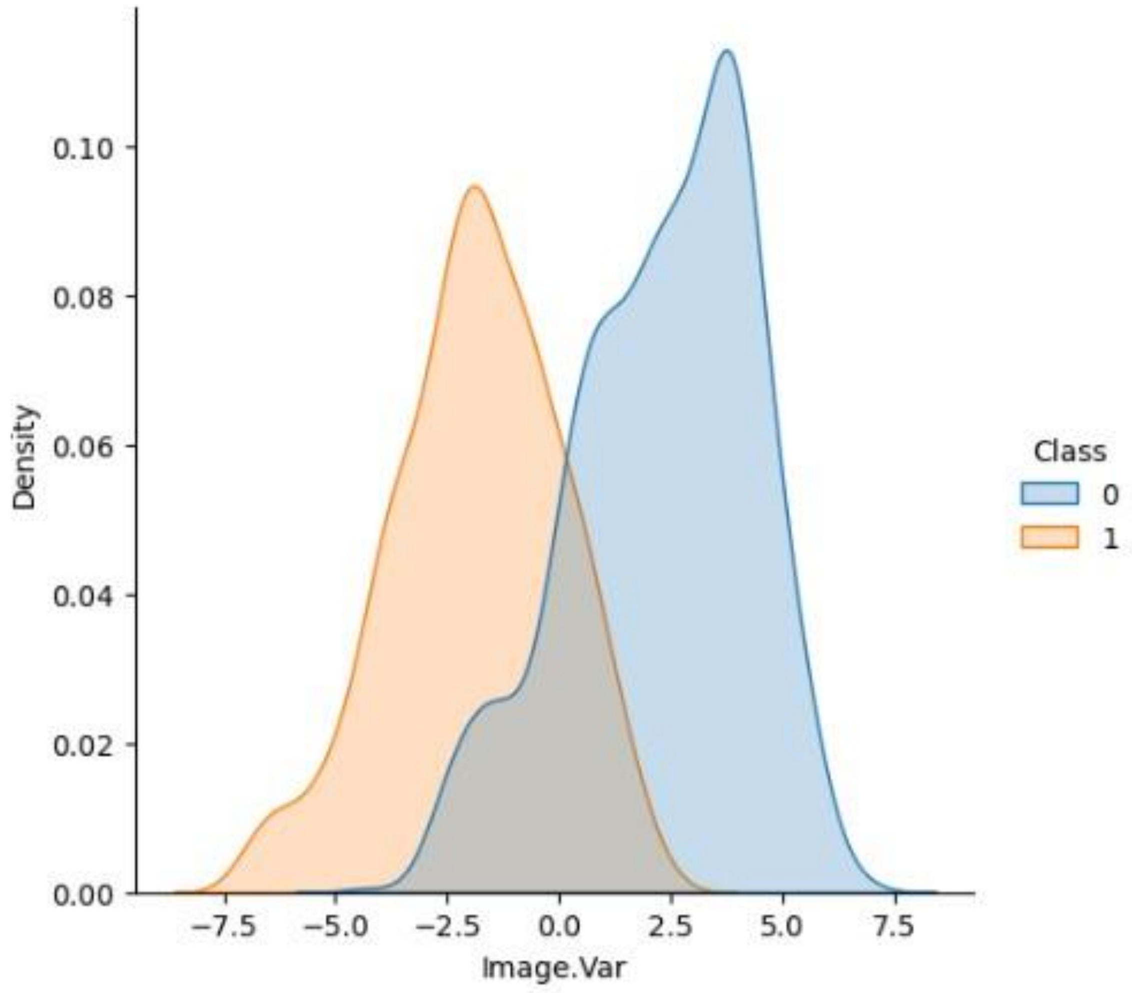
Importing Libraries

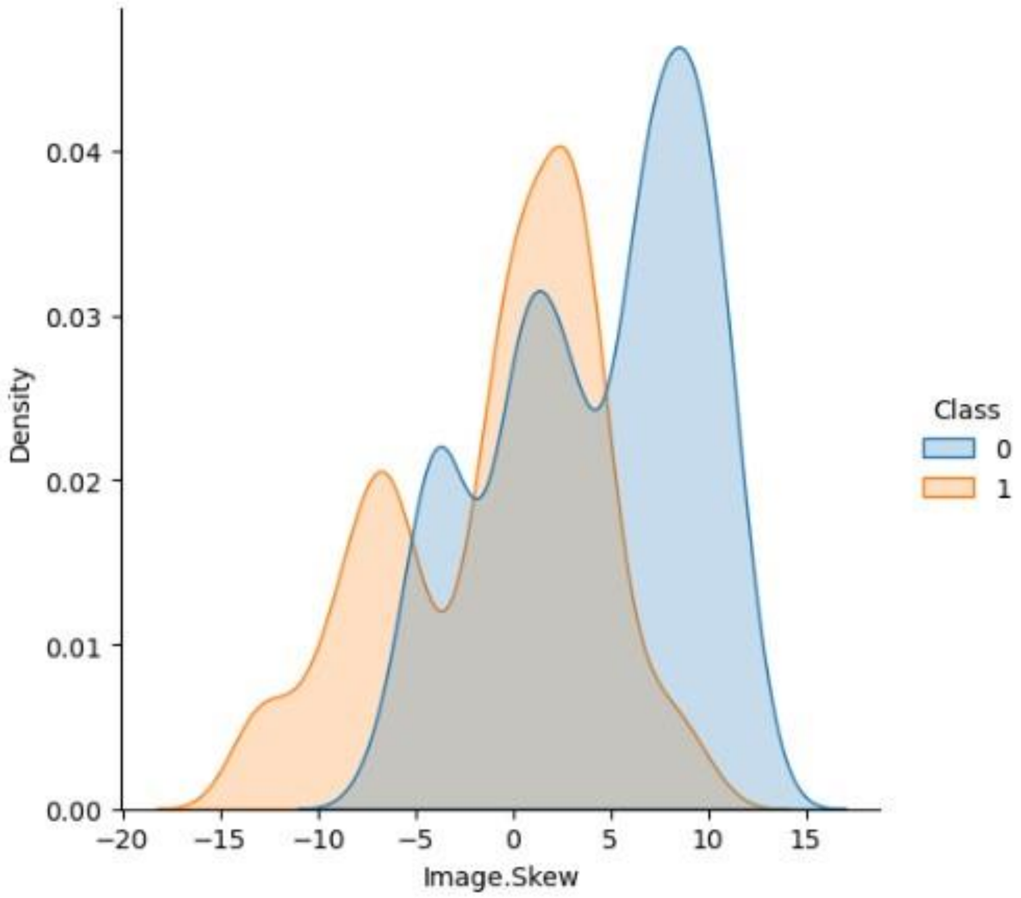


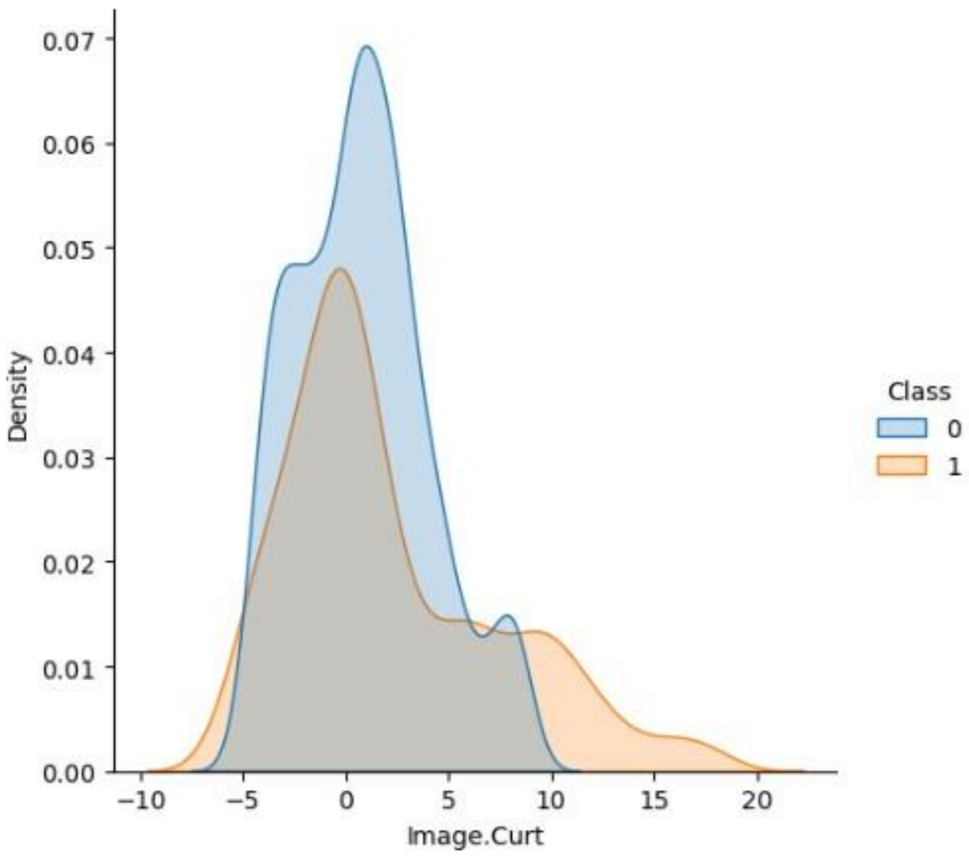
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('banknote.csv')

# Check the distribution of each variable
sns.displot(data=df, x="Image.Var", hue="Class", kind="kde", fill=True)
sns.displot(data=df, x="Image.Skew", hue="Class", kind="kde", fill=True)
sns.displot(data=df, x="Image.Curt", hue="Class", kind="kde", fill=True)
sns.displot(data=df, x="Entropy", hue="Class", kind="kde", fill=True)
```

Checking the correlation matrix of the variables

```python
# Check the correlation matrix of the variables
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```



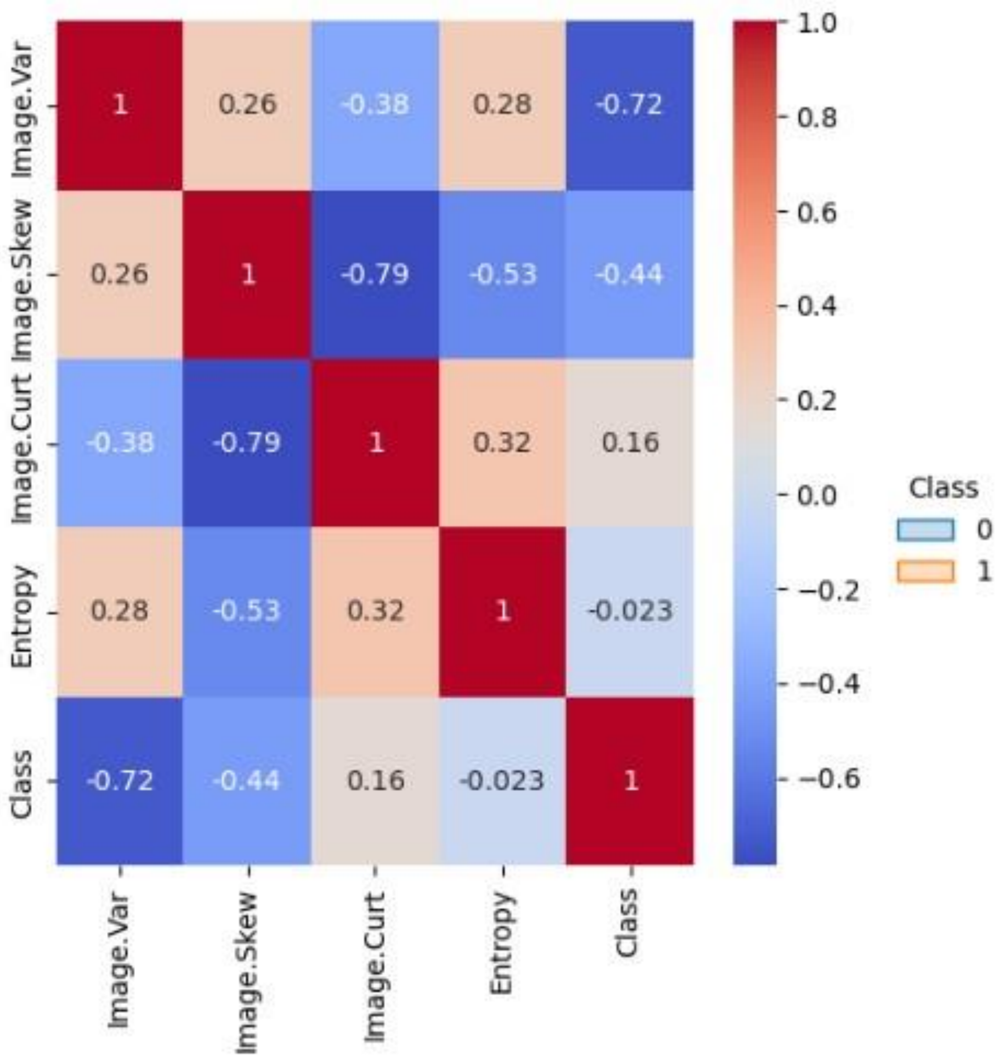

Density
0.00
0.01
0.02
0.03
0.04
−20
−15
−10
−5
0
5
10
15
Image.Skew
Class
0
1



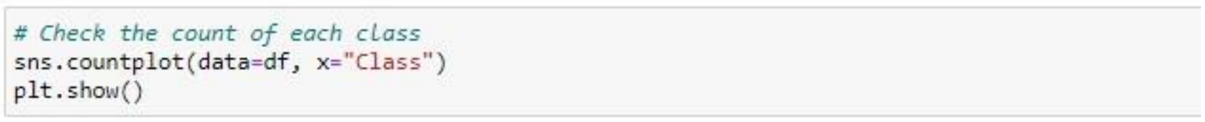
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
−10
−5
0
5
10
15
20
Image.Curt
Class
0
1

Heat Map

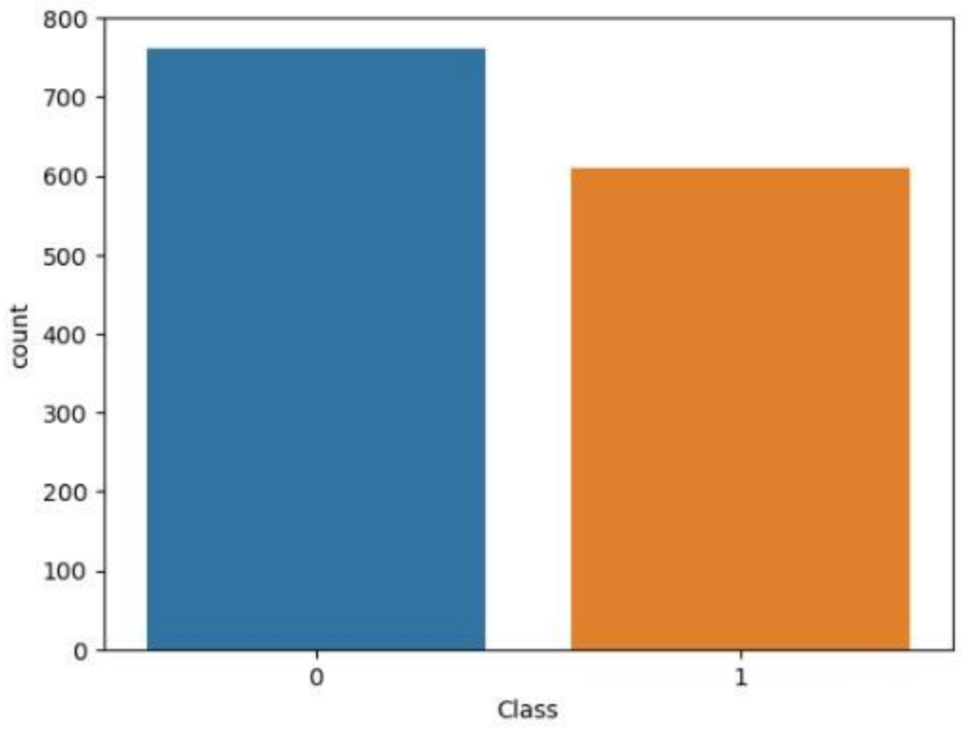
Finding more information about the dataset.



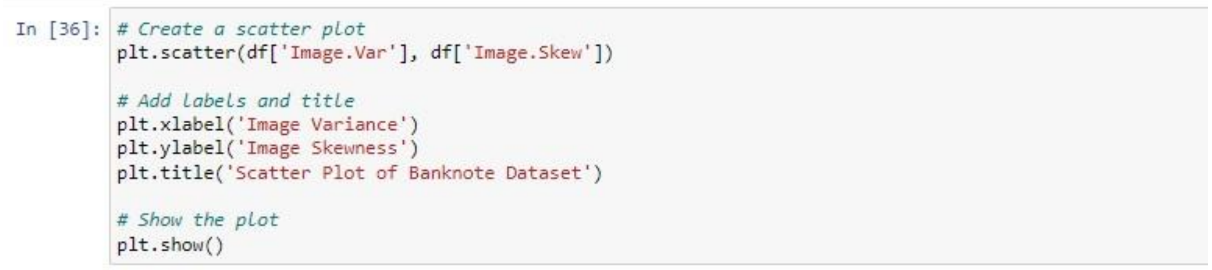
Checking the count of each class



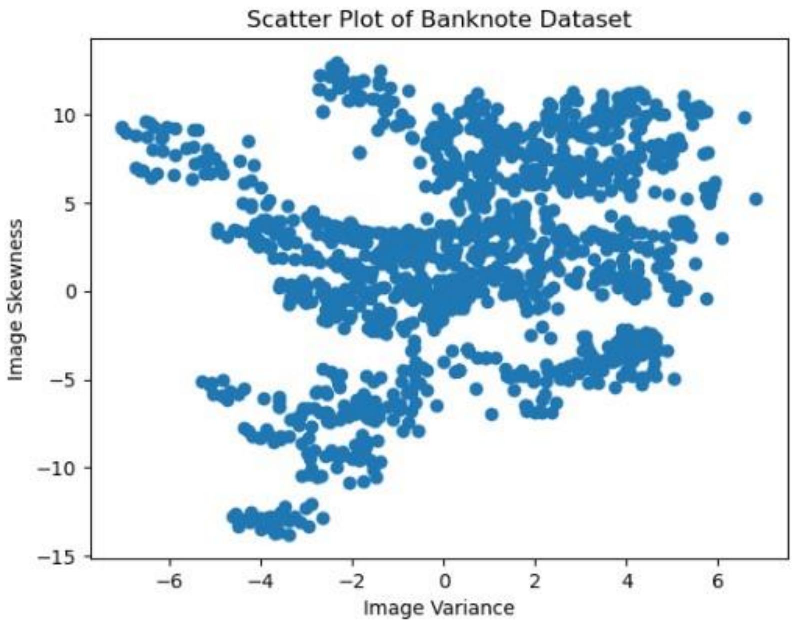
```python
# Check the count of each class
sns.countplot(data=df, x="Class")
plt.show()
```



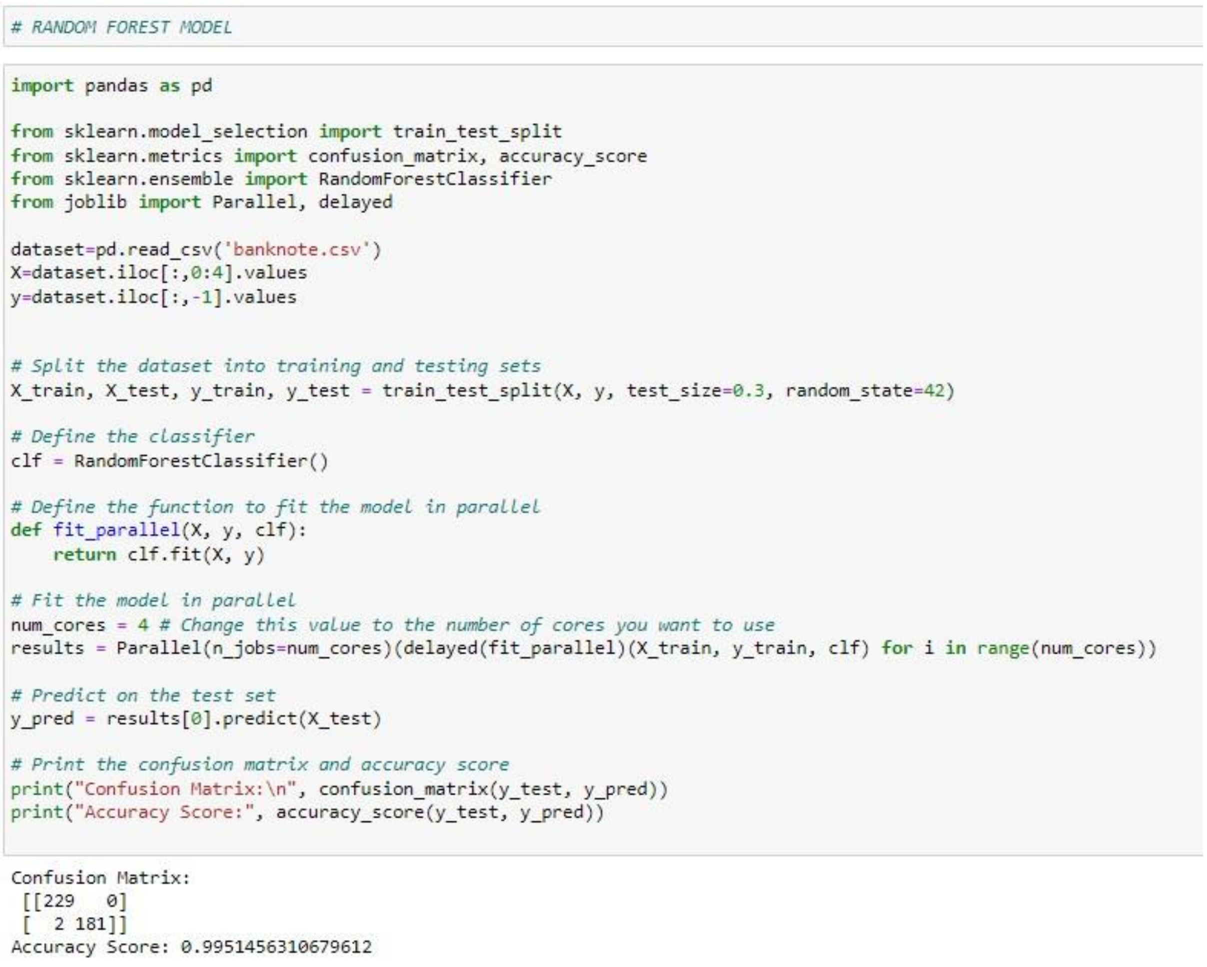
Creating a scatter plot



```
In [36]: # Create a scatter plot
         plt.scatter(df['Image.Var'], df['Image.Skew'])

         # Add labels and title
         plt.xlabel('Image Variance')
         plt.ylabel('Image Skewness')
         plt.title('Scatter Plot of Banknote Dataset')

         # Show the plot
         plt.show()
```



1. Random Forest Model



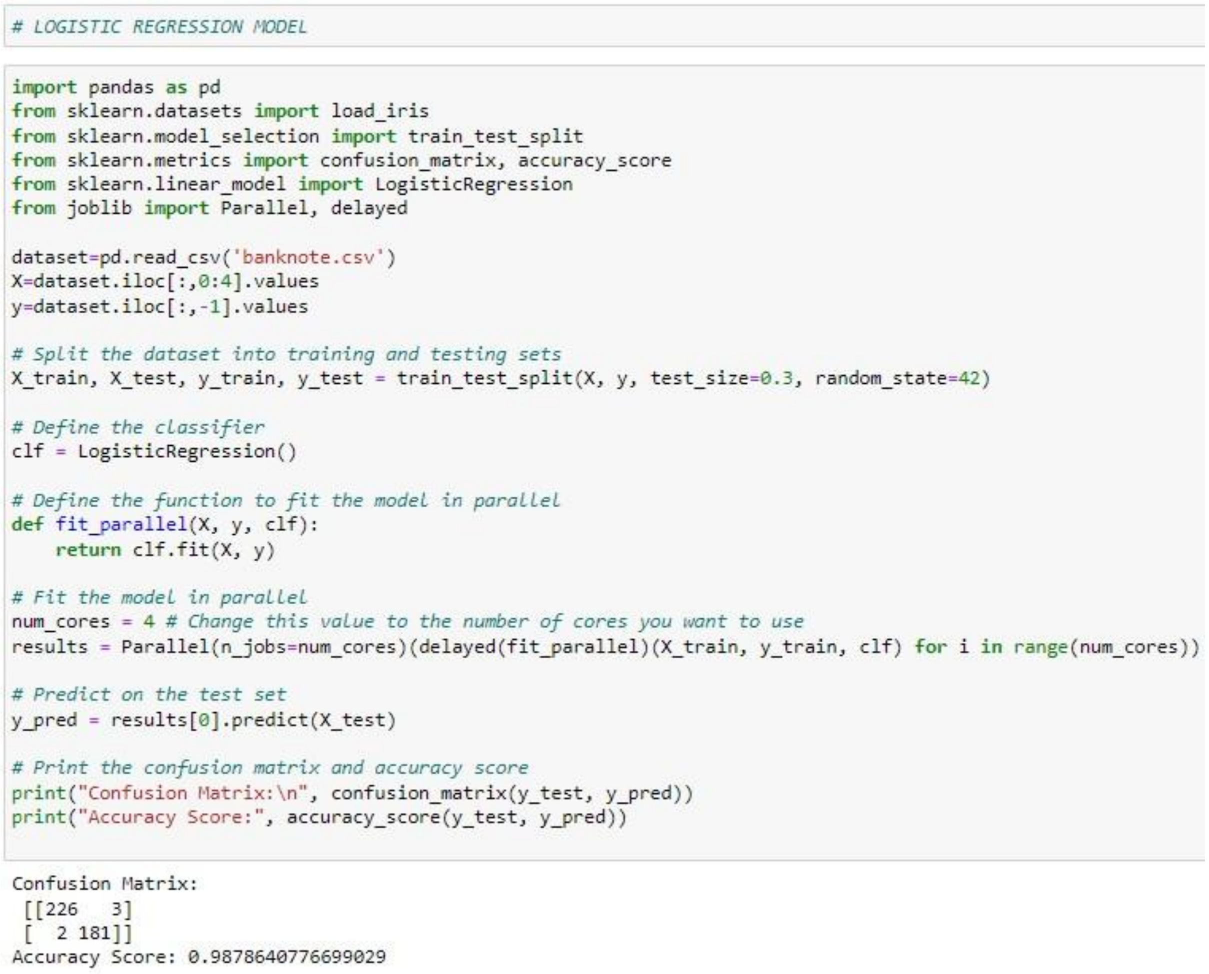
```
# RANDOM FOREST MODEL
```

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from joblib import Parallel, delayed

dataset=pd.read_csv('banknote.csv')
X=dataset.iloc[:,0:4].values
y=dataset.iloc[:,-1].values


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define the classifier
clf = RandomForestClassifier()

# Define the function to fit the model in parallel
def fit_parallel(X, y, clf):
    return clf.fit(X, y)

# Fit the model in parallel
num_cores = 4 # Change this value to the number of cores you want to use
results = Parallel(n_jobs=num_cores)(delayed(fit_parallel)(X_train, y_train, clf) for i in range(num_cores))

# Predict on the test set
y_pred = results[0].predict(X_test)

# Print the confusion matrix and accuracy score
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[229   0]
 [  2 181]]
Accuracy Score: 0.9951456310679612
```

2.

Logistic Regression Model

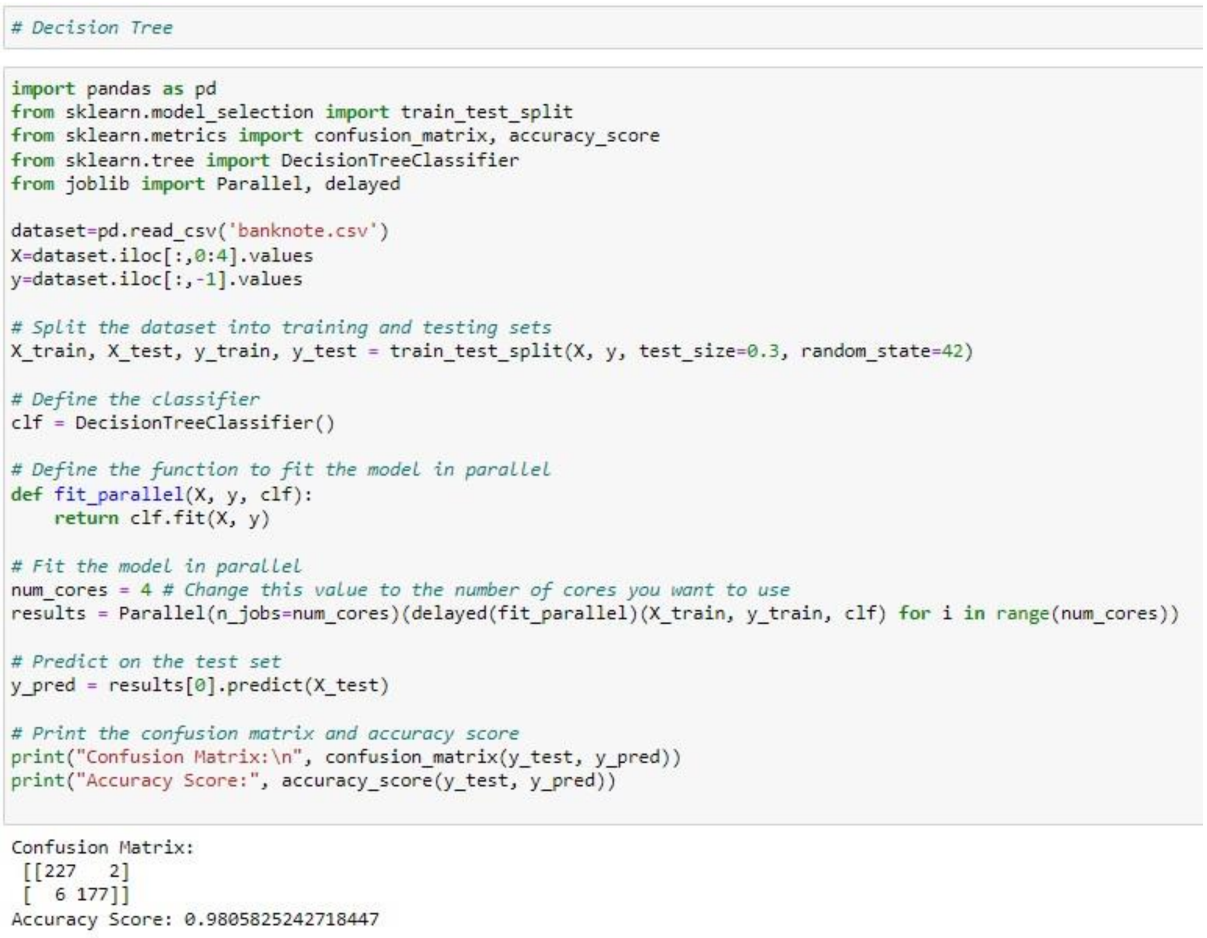


```
# LOGISTIC REGRESSION MODEL
```

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression
from joblib import Parallel, delayed

dataset=pd.read_csv('banknote.csv')
X=dataset.iloc[:,0:4].values
y=dataset.iloc[:,-1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define the classifier
clf = LogisticRegression()

# Define the function to fit the model in parallel
def fit_parallel(X, y, clf):
    return clf.fit(X, y)

# Fit the model in parallel
num_cores = 4 # Change this value to the number of cores you want to use
results = Parallel(n_jobs=num_cores)(delayed(fit_parallel)(X_train, y_train, clf) for i in range(num_cores))

# Predict on the test set
y_pred = results[0].predict(X_test)

# Print the confusion matrix and accuracy score
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[226   3]
 [  2 181]]
Accuracy Score: 0.9878640776699029
```

3. Decision Tree



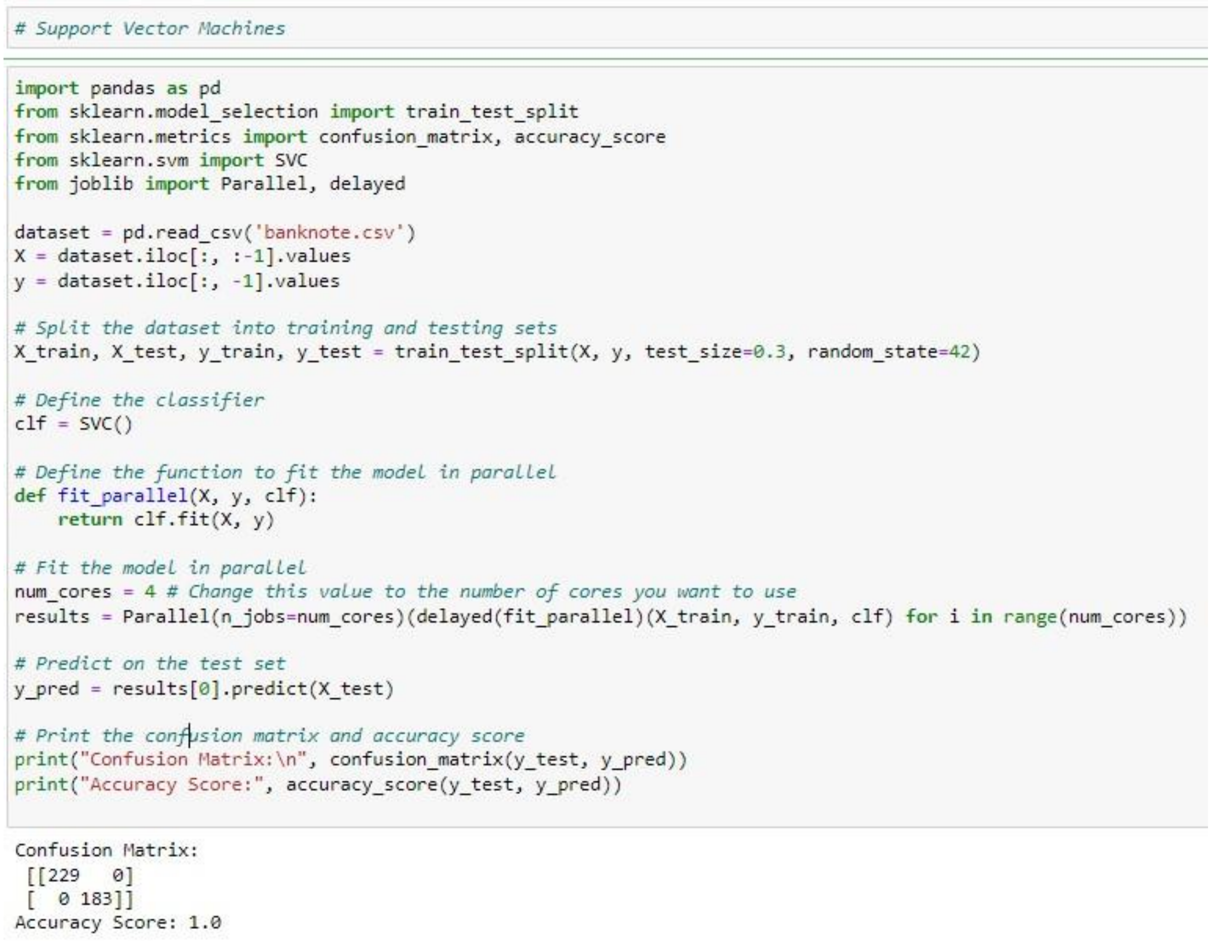
```
# Decision Tree
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from joblib import Parallel, delayed

dataset=pd.read_csv('banknote.csv')
X=dataset.iloc[:,0:4].values
y=dataset.iloc[:,-1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define the classifier
clf = DecisionTreeClassifier()

# Define the function to fit the model in parallel
def fit_parallel(X, y, clf):
    return clf.fit(X, y)

# Fit the model in parallel
num_cores = 4 # Change this value to the number of cores you want to use
results = Parallel(n_jobs=num_cores)(delayed(fit_parallel)(X_train, y_train, clf) for i in range(num_cores))

# Predict on the test set
y_pred = results[0].predict(X_test)

# Print the confusion matrix and accuracy score
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[227   2]
 [  6 177]]
Accuracy Score: 0.9805825242718447
```

4.

Support Vector Machine

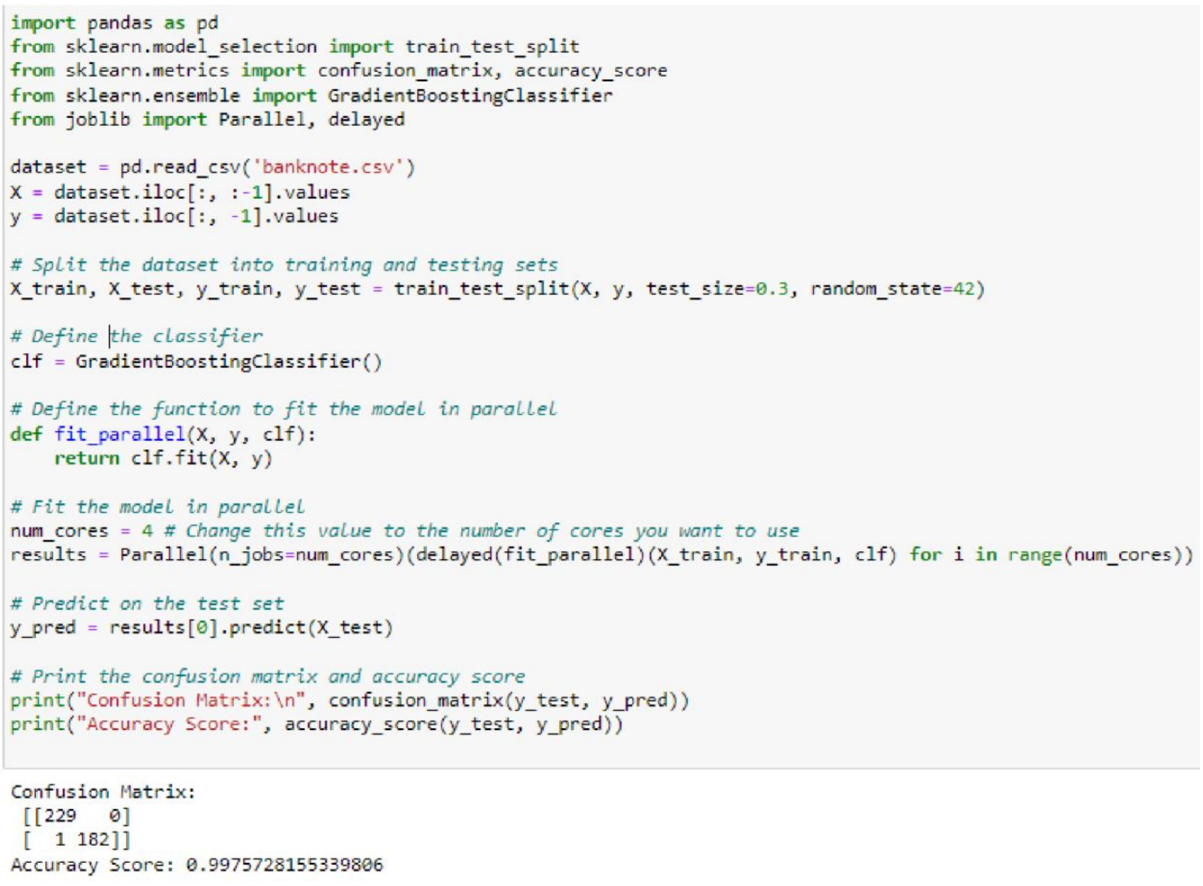


```
# Support Vector Machines
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.svm import SVC
from joblib import Parallel, delayed

dataset = pd.read_csv('banknote.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define the classifier
clf = SVC()

# Define the function to fit the model in parallel
def fit_parallel(X, y, clf):
    return clf.fit(X, y)

# Fit the model in parallel
num_cores = 4 # Change this value to the number of cores you want to use
results = Parallel(n_jobs=num_cores)(delayed(fit_parallel)(X_train, y_train, clf) for i in range(num_cores))

# Predict on the test set
y_pred = results[0].predict(X_test)

# Print the confusion matrix and accuracy score
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[229   0]
 [  0 183]]
Accuracy Score: 1.0
```

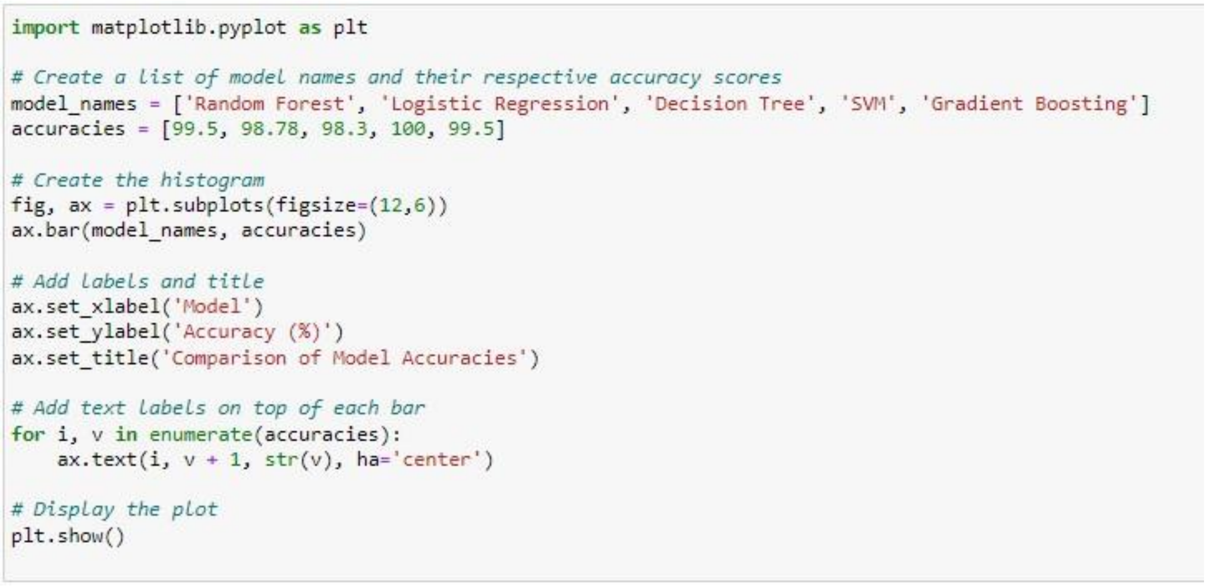
5. Gradient Boosting



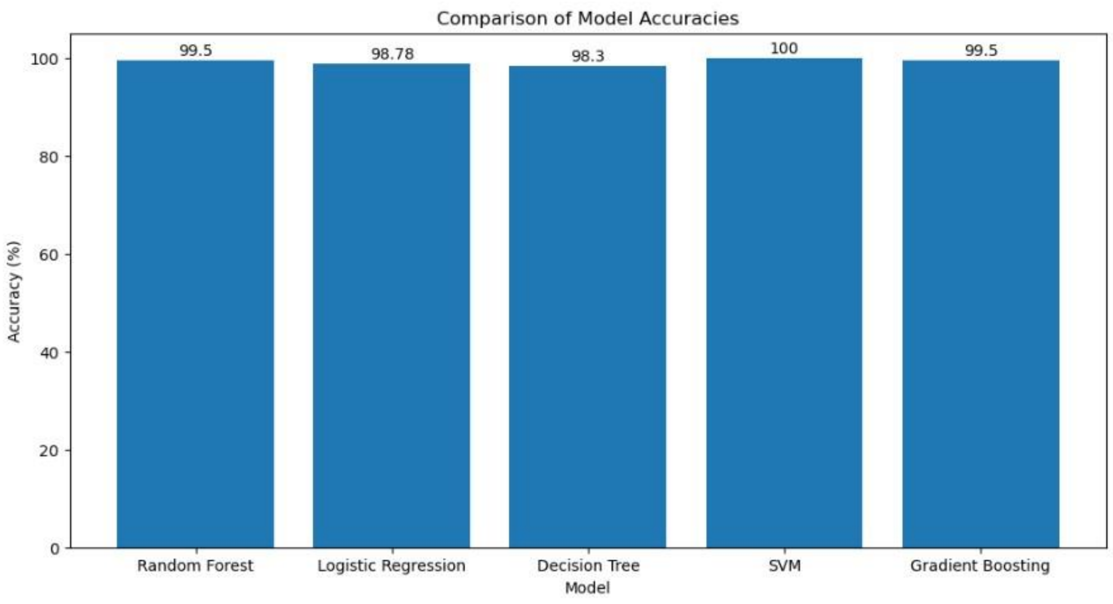
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.ensemble import GradientBoostingClassifier
from joblib import Parallel, delayed

dataset = pd.read_csv('banknote.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define the classifier
clf = GradientBoostingClassifier()

# Define the function to fit the model in parallel
def fit_parallel(X, y, clf):
    return clf.fit(X, y)

# Fit the model in parallel
num_cores = 4 # Change this value to the number of cores you want to use
results = Parallel(n_jobs=num_cores)(delayed(fit_parallel)(X_train, y_train, clf) for i in range(num_cores))

# Predict on the test set
y_pred = results[0].predict(X_test)

# Print the confusion matrix and accuracy score
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[229   0]
 [  1 182]]
Accuracy Score: 0.9975728155339806
```

RESULTS:



```
import matplotlib.pyplot as plt

# Create a list of model names and their respective accuracy scores
model_names = ['Random Forest', 'Logistic Regression', 'Decision Tree', 'SVM', 'Gradient Boosting']
accuracies = [99.5, 98.78, 98.3, 100, 99.5]

# Create the histogram
fig, ax = plt.subplots(figsize=(12,6))
ax.bar(model_names, accuracies)

# Add labels and title
ax.set_xlabel('Model')
ax.set_ylabel('Accuracy (%)')
ax.set_title('Comparison of Model Accuracies')

# Add text labels on top of each bar
for i, v in enumerate(accuracies):
    ax.text(i, v + 1, str(v), ha='center')

# Display the plot
plt.show()
```



Conclusion:

All in all, cash transactions might be a big challenge faced in terms of counterfeit or fake notes and as it get worse due to the advancements in technology the only way to counter it is technology. Our work with the ML models used does exactly that and opens up future scope of work to add more models to increase the accuracy of the prediction model. Counterfeit or fake notes are an issue of concern to almost every country but technology can be used to avoid the consequences of Counterfeit notes being used.

Demonstration Link - https://vitacin.sharepoint.com/:v:/s/PDCRev-3/EQ4xvMpTolNAs3Y4wj617ScBTG6Wlk5TZU7VKEXOJjqgig?e=A9xizw