BANK NOTE AUTHENTICATION

ADVANCED MACHINE LEARNING

BY JITHIN JAYACHANDRAN

INTRODUCTION

PROJECT OBJECTIVE:

DEVELOP A MACHINE LEARNING MODEL TO AUTHENTICATE BANKNOTES BY CLASSIFYING THEM AS GENUINE OR FORGED.

PROBLEM STATEMENT:

THE RISE OF COUNTERFEIT CURRENCY PRESENTS A SIGNIFICANT THREAT TO FINANCIAL SYSTEMS WORLDWIDE.
AN AUTOMATED, ACCURATE METHOD FOR DISTINGUISHING BETWEEN GENUINE AND FORGED BANKNOTES IS CRUCIAL.

DATASET OVERVIEW:

SOURCE: DATA EXTRACTED FROM HIGH-RESOLUTION IMAGES OF BANKNOTES USING AN INDUSTRIAL CAMERA TYPICALLY EMPLOYED FOR PRINT INSPECTION.
FEATURES: THE DATASET CONTAINS 1372 INSTANCES WITH 4 CONTINUOUS FEATURES: VARIANCE,SKEWNESS,CURTOSIS,ENTROPY

ASSOCIATED TASK:

CLASSIFICATION OF BANKNOTES BASED ON THE EXTRACTED FEATURES TO DETERMINE THEIR AUTHENTICITY.

IMPORTANCE OF BANKNOTE AUTHENTICATION:

COMBATTING COUNTERFEITING: COUNTERFEIT CURRENCY LEADS TO SIGNIFICANT FINANCIAL LOSSES AND DIMINISHES TRUST IN THE MONETARY SYSTEM.
CHALLENGES IN MANUAL DETECTION: TRADITIONAL METHODS ARE SLOW, ERROR-PRONE, AND INCREASINGLY INEFFECTIVE AGAINST SOPHISTICATED FORGERIES.
ROLE OF MACHINE LEARNING: MACHINE LEARNING PROVIDES A POWERFUL, AUTOMATED SOLUTION FOR ACCURATELY DETECTING COUNTERFEIT BANKNOTES, OFFERING A SCALABLE AND RELIABLE ALTERNATIVE TO MANUAL CHECKS.

DATASET OVERVIEW

DATASET NAME: BANKNOTE AUTHENTICATION DATASET

SOURCE:
DESCRIPTION: DATA EXTRACTED FROM HIGH-RESOLUTION IMAGES OF BANKNOTES.
CAMERA USED: INDUSTRIAL CAMERA TYPICALLY EMPLOYED FOR PRINT INSPECTION.

DATASET CHARACTERISTICS:
INSTANCES: 1372
FEATURES: 4 CONTINUOUS FEATURES
TARGET VARIABLE: 1 INTEGER CLASS LABEL

FEATURE OVERVIEW:
1. VARIANCE:
TYPE: CONTINUOUS
DESCRIPTION: MEASURES THE VARIANCE OF THE PIXEL VALUES IN THE IMAGE.
2. SKEWNESS:
TYPE: CONTINUOUS
DESCRIPTION: MEASURES THE ASYMMETRY OF THE PIXEL VALUE DISTRIBUTION IN THE IMAGE.
3. CURTOSIS:
TYPE: CONTINUOUS
DESCRIPTION: MEASURES THE PEAKEDNESS OF THE PIXEL VALUE DISTRIBUTION IN THE IMAGE.
4. ENTROPY:
TYPE: CONTINUOUS
DESCRIPTION: MEASURES THE RANDOMNESS OR COMPLEXITY OF THE PIXEL VALUES IN THE IMAGE.
5. CLASS:
TYPE: INTEGER
DESCRIPTION: INDICATES WHETHER THE BANKNOTE IS GENUINE (0) OR FORGED (1).

DATA SUMMARY:
MISSING VALUES: NONE
DUPLICATES: CHECKED AND REMOVED IF PRESENT
RANGE OF VALUES: FEATURES HAVE VARIED RANGES DEPENDING ON THE SCALE OF THE PIXEL VALUES.

LOAD THE DATASET

```
import pandas as pd
column_names = ["variance", "skewness", "curtosis", "entropy", "class"]
df = pd.read_csv('/content/data_banknote_authentication.txt',header=None, names=column_names)
df.head()
```

	variance	skewness	curtosis	entropy	class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

EDA & PREPROCESSING

```
# Check for missing values
df.isnull().sum()
```

	0
variance	0
skewness	0
curtosis	0
entropy	0
class	0

dtype: int64

```
[3] # Summary of Dataset
    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
```

```
[4] # Check for duplicate values
    duplicates = df.duplicated().sum()
    print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 24
```

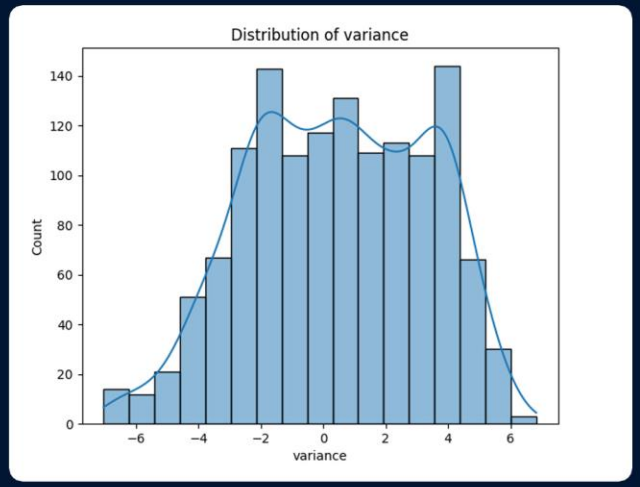
```
[5] # Remove duplicate values
    df = df.drop_duplicates()
    print(f"Duplicate rows removed. Number of rows after removal: {len(df)}")
```

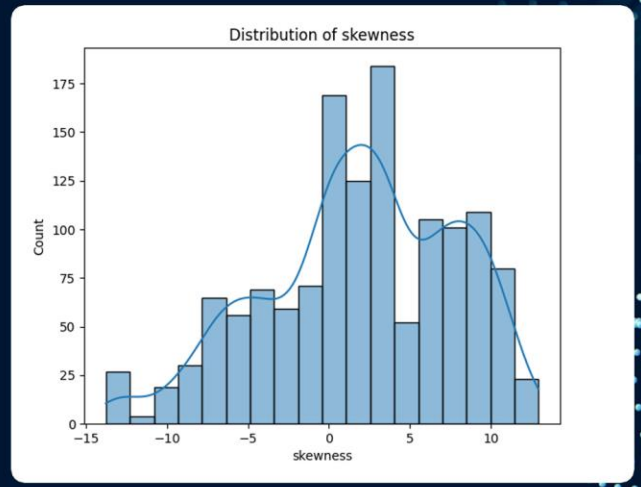
```
Duplicate rows removed. Number of rows after removal: 1348
```

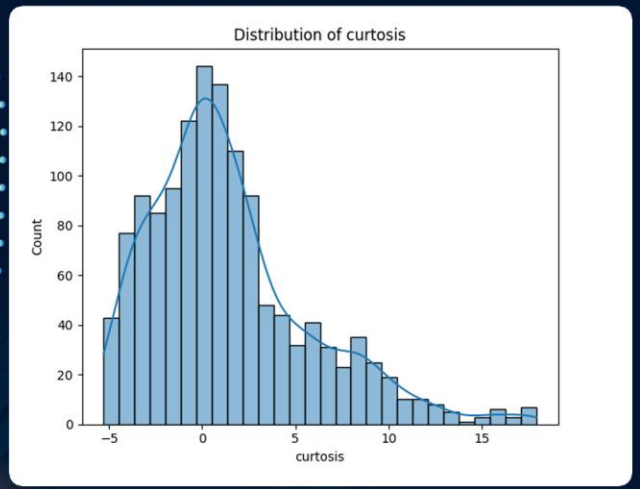
```
[6] # Summary statistics
    print("\nSummary statistics:")
    print(df.describe())
```

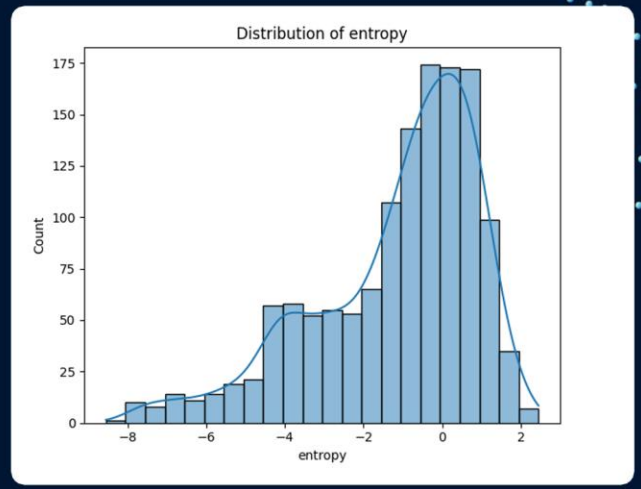
```
Summary statistics:
          variance     skewness     curtosis      entropy        class
count  1348.000000  1348.000000  1348.000000  1348.000000  1348.000000
mean      0.445785     1.909039     1.413578    -1.168712     0.452522
std       2.862906     5.868600     4.328365     2.085877     0.497925
min      -7.042100   -13.773100    -5.286100    -8.548200     0.000000
25%      -1.786650    -1.627000    -1.545600    -2.393100     0.000000
50%       0.518735     2.334150     0.605495    -0.578890     0.000000
75%       2.853250     6.796025     3.199800     0.403863     1.000000
max       6.824800    12.951600    17.927400     2.449500     1.000000
```

VISUALIZE THE DISTRIBUTION OF EACH FEATURE

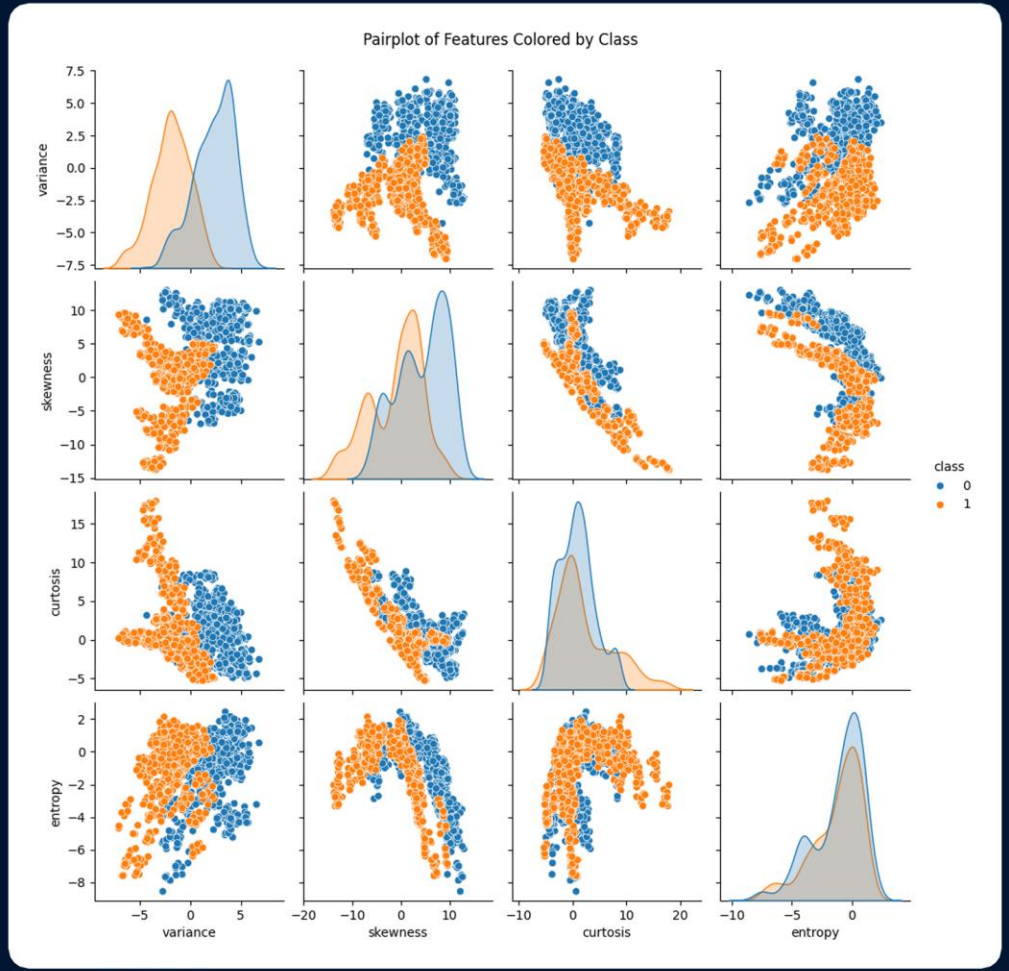




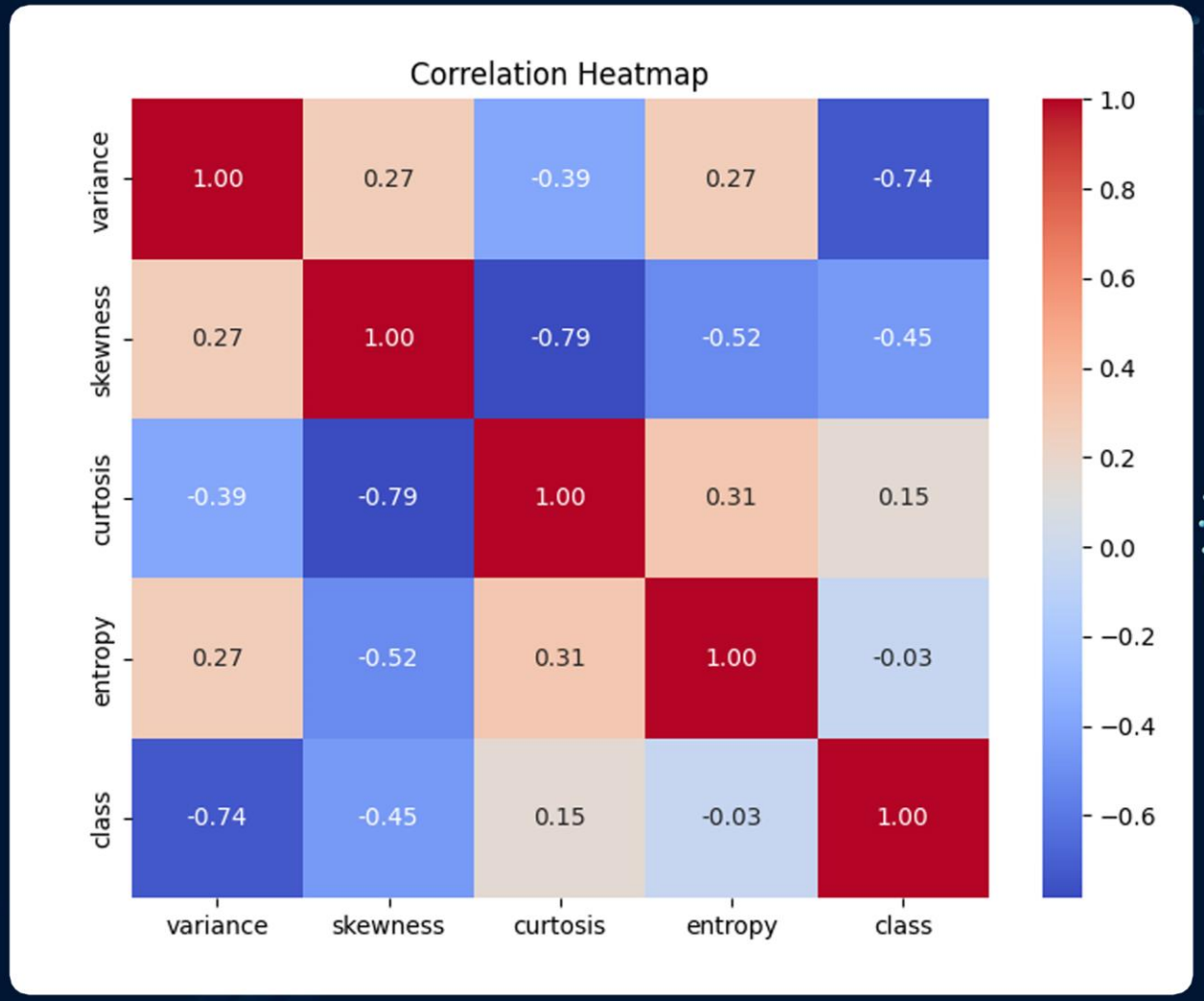




VISUALIZE THE PAIRWISE RELATIONSHIPS BETWEEN FEATURES



CORRELATION HEATMAP



HANDLING OUTLIERS

```
[11] import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from scipy import stats

     # Data Cleaning: Handling outliers using Z-Score
     z_scores = np.abs(stats.zscore(df.iloc[:, :-1]))
     outliers = np.where(z_scores > 3, True, False)
     print(f"\nNumber of outliers detected: {np.sum(outliers)}")
     df = df[(z_scores < 3).all(axis=1)]
     print(f"Number of rows after removing outliers: {len(df)}")
```

```
Number of outliers detected: 34
Number of rows after removing outliers: 1314
```

MODEL SELECTION

```
[15] from sklearn.model_selection import cross_val_score, GridSearchCV
     from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
     from sklearn.svm import SVC
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
[16] # Model Training and Selection
     models = {
         'Random Forest': RandomForestClassifier(random_state=42),
         'Gradient Boosting': GradientBoostingClassifier(random_state=42),
         'SVM': SVC(random_state=42),
         'K-Nearest Neighbors': KNeighborsClassifier(),
         'Logistic Regression': LogisticRegression(random_state=42),
         'Decision Tree': DecisionTreeClassifier(random_state=42)
     }
```

```python
[17] best_model = None
     best_accuracy = 0

     for model_name, model in models.items():
         print(f"\nTraining and evaluating {model_name}...")
         cv_scores = cross_val_score(model, xtrain, ytrain, cv=5, scoring='accuracy')
         mean_cv_score = cv_scores.mean()
         print(f"Cross-validation scores: {cv_scores}")
         print(f"Mean CV accuracy: {mean_cv_score}")

         # Train the model
         model.fit(xtrain, ytrain)
         trainpred = model.predict(xtrain)
         testpred = model.predict(xtest)

         # Evaluate the model
         train_accuracy = accuracy_score(ytrain, trainpred)
         test_accuracy = accuracy_score(ytest, testpred)

         print(f"Training Accuracy: {train_accuracy}")
         print(f"Testing Accuracy: {test_accuracy}")

         # Select the best model based on test accuracy
         if test_accuracy > best_accuracy:
             best_accuracy = test_accuracy
             best_model = model_name

     print(f"\nBest model based on testing accuracy: {best_model} with accuracy {best_accuracy}")
```

```
Training and evaluating Random Forest...
Cross-validation scores: [0.98578199 0.9952381  0.9952381  1.         0.98571429]
Mean CV accuracy: 0.9923944933423605
Training Accuracy: 1.0
Testing Accuracy: 0.9961977186311787

Training and evaluating Gradient Boosting...
Cross-validation scores: [0.98578199 0.99047619 1.         0.9952381  0.99047619]
Mean CV accuracy: 0.9923944933423605
Training Accuracy: 1.0
Testing Accuracy: 0.9923954372623575

Training and evaluating SVM...
Cross-validation scores: [0.99526066 1.         0.9952381  1.         1.        ]
Mean CV accuracy: 0.9980997517490409
Training Accuracy: 0.9980970504281637
Testing Accuracy: 1.0

Training and evaluating K-Nearest Neighbors...
Cross-validation scores: [0.99052133 1.         0.99047619 1.         0.9952381 ]
Mean CV accuracy: 0.9952471225457007
Training Accuracy: 0.9961941008563273
Testing Accuracy: 1.0

Training and evaluating Logistic Regression...
Cross-validation scores: [0.98578199 0.9952381  0.9952381  0.99047619 0.9952381 ]
Mean CV accuracy: 0.9923944933423605
Training Accuracy: 0.9933396764985728
Testing Accuracy: 0.9885931558935361

Training and evaluating Decision Tree...
Cross-validation scores: [0.99052133 0.9952381  0.97142857 0.99047619 0.98095238]
Mean CV accuracy: 0.9857233130218912
Training Accuracy: 1.0
Testing Accuracy: 0.9771863117870723

Best model based on testing accuracy: SVM with accuracy 1.0
```

TRAIN THE MODEL

```
# Evaluate the best model
print("\nFinal Model Evaluation:")
model = models[best_model]
model.fit(xtrain, ytrain)
```

```
Final Model Evaluation:
```

SVC

```
SVC(random_state=42)
```

GET PREDICTIONS

```
trainpred = model.predict(xtrain)
testpred = model.predict(xtest)
```

EVALUATE THE MODEL

```
[83] # Evaluate the model on the training set
     print("\nEvaluation on Training Set:")
     print("Confusion Matrix:")
     print(confusion_matrix(ytrain, trainpred))
     print("\nClassification Report:")
     print(classification_report(ytrain, trainpred))
     print("Accuracy Score:")
     print(accuracy_score(ytrain, trainpred))
```

```
Evaluation on Training Set:
Confusion Matrix:
[[575   2]
 [  0 474]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       577
           1       1.00      1.00      1.00       474

    accuracy                           1.00      1051
   macro avg       1.00      1.00      1.00      1051
weighted avg       1.00      1.00      1.00      1051

Accuracy Score:
0.9980970504281637
```

```
[84] # Evaluate the model on the test set
     print("\nEvaluation on Test Set:")
     print("Confusion Matrix:")
     print(confusion_matrix(ytest, testpred))
     print("\nClassification Report:")
     print(classification_report(ytest, testpred))
     print("Accuracy Score:")
     print(accuracy_score(ytest, testpred))
```

```
Evaluation on Test Set:
Confusion Matrix:
[[151   0]
 [  0 112]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       151
           1       1.00      1.00      1.00       112

    accuracy                           1.00       263
   macro avg       1.00      1.00      1.00       263
weighted avg       1.00      1.00      1.00       263

Accuracy Score:
1.0
```

Training Set vs. Test Set

Metric	Training Set	Test Set
Confusion Matrix	\[5752 0474]	\[1510 0112]
	- True Negatives (TN): 575 (Class 0 correctly predicted as Class 0)	- True Negatives (TN): 151 (Class 0 correctly predicted as Class 0)
	- False Positives (FP): 2 (Class 0 incorrectly predicted as Class 1)	- False Positives (FP): 0 (No Class 0 incorrectly predicted as Class 1)
	- False Negatives (FN): 0 (No Class 1 incorrectly predicted as Class 0)	- False Negatives (FN): 0 (No Class 1 incorrectly predicted as Class 0)
	- True Positives (TP): 474 (Class 1 correctly predicted as Class 1)	- True Positives (TP): 112 (Class 1 correctly predicted as Class 1)
Precision	1.00 (100%) for both classes	1.00 (100%) for both classes
	- Class 0: 1.00 (All predicted Class 0 were actual Class 0)	- Class 0: 1.00 (All predicted Class 0 were actual Class 0)
	- Class 1: 1.00 (All predicted Class 1 were actual Class 1)	- Class 1: 1.00 (All predicted Class 1 were actual Class 1)
Recall	1.00 (100%) for both classes	1.00 (100%) for both classes
	- Class 0: 1.00 (All actual Class 0 were correctly predicted)	- Class 0: 1.00 (All actual Class 0 were correctly predicted)
	- Class 1: 1.00 (All actual Class 1 were correctly predicted)	- Class 1: 1.00 (All actual Class 1 were correctly predicted)
F1-Score	1.00 (100%) for both classes	1.00 (100%) for both classes
	- Class 0: 1.00 (Perfect balance between precision and recall)	- Class 0: 1.00 (Perfect balance between precision and recall)
	- Class 1: 1.00 (Perfect balance between precision and recall)	- Class 1: 1.00 (Perfect balance between precision and recall)
Support	Class 0: 577, Class 1: 474	Class 0: 151, Class 1: 112
	- Class 0: 577 instances in training set	- Class 0: 151 instances in test set
	- Class 1: 474 instances in training set	- Class 1: 112 instances in test set
Accuracy Score	0.9981 (99.81%)	1.0 (100%)
	- The model correctly predicted 99.81% of the training instances.	- The model correctly predicted 100% of the test instances.
Macro Average	Precision: 1.00, Recall: 1.00, F1-Score: 1.00	Precision: 1.00, Recall: 1.00, F1-Score: 1.00
	- Average of metrics across all classes, treating all classes equally.	- Average of metrics across all classes, treating all classes equally.
Weighted Average	Precision: 1.00, Recall: 1.00, F1-Score: 1.00	Precision: 1.00, Recall: 1.00, F1-Score: 1.00
	- Average of metrics weighted by the number of instances in each class.	- Average of metrics weighted by the number of instances in each class.

SUMMARY

THE MACHINE LEARNING MODEL, SPECIFICALLY THE SUPPORT VECTOR CLASSIFIER (SVC), HAS DEMONSTRATED EXCEPTIONAL PERFORMANCE IN CLASSIFYING BANKNOTES AS EITHER GENUINE OR FORGED. THE RESULTS FROM BOTH THE TRAINING AND TEST SETS REVEAL THE FOLLOWING:

PERFECT CLASSIFICATION ACCURACY: THE MODEL ACHIEVED NEARLY PERFECT ACCURACY ON THE TRAINING SET (99.81%) AND PERFECT ACCURACY ON THE TEST SET (100%). THIS INDICATES THAT THE MODEL CAN RELIABLY IDENTIFY BOTH GENUINE AND FORGED BANKNOTES.

PRECISION, RECALL, AND F1-SCORE:

PRECISION: THE MODEL ACHIEVED A PRECISION SCORE OF 1.00 (100%) FOR BOTH CLASSES IN BOTH TRAINING AND TEST SETS, MEANING IT CORRECTLY IDENTIFIES ALL POSITIVE CASES WITHOUT FALSE POSITIVES.
RECALL: WITH A RECALL SCORE OF 1.00 (100%) FOR BOTH CLASSES, THE MODEL ACCURATELY DETECTS ALL ACTUAL POSITIVE CASES WITHOUT MISSING ANY.
F1-SCORE: THE F1-SCORE, WHICH BALANCES PRECISION AND RECALL, IS ALSO 1.00 (100%) FOR BOTH CLASSES, INDICATING AN OPTIMAL BALANCE BETWEEN PRECISION AND RECALL.
CONFUSION MATRIX ANALYSIS: THE CONFUSION MATRICES FOR BOTH TRAINING AND TEST SETS SHOW NO FALSE POSITIVES OR FALSE NEGATIVES, REFLECTING THE MODEL'S HIGH RELIABILITY AND ACCURACY IN DISTINGUISHING BETWEEN GENUINE AND FORGED BANKNOTES.

REAL-WORLD IMPLICATIONS:

SCALABILITY: GIVEN ITS HIGH ACCURACY AND ROBUSTNESS, THE MODEL IS WELL-SUITED FOR REAL-WORLD APPLICATIONS WHERE ACCURATE BANKNOTE AUTHENTICATION IS CRITICAL. IT CAN BE EFFECTIVELY DEPLOYED IN ENVIRONMENTS REQUIRING HIGH PRECISION AND RELIABILITY.
RELIABILITY: THE MODEL'S PERFORMANCE ON UNSEEN TEST DATA SUGGESTS THAT IT CAN GENERALIZE WELL TO NEW, UNSEEN BANKNOTES, MAKING IT A ROBUST SOLUTION FOR PRACTICAL USE.

FUTURE CONSIDERATIONS:

ADAPTABILITY: ALTHOUGH THE MODEL PERFORMS EXCELLENTLY WITH THE CURRENT DATASET, ONGOING MONITORING AND RETRAINING WITH NEW DATA MAY BE NECESSARY TO ADAPT TO POTENTIAL CHANGES IN COUNTERFEIT TECHNIQUES OR VARIATIONS IN BANKNOTE DESIGNS.
INTEGRATION: THE MODEL CAN BE INTEGRATED INTO AUTOMATED BANKNOTE VERIFICATION SYSTEMS, ENHANCING THE SECURITY AND EFFICIENCY OF FINANCIAL TRANSACTIONS.

IN SUMMARY, THE SUPPORT VECTOR CLASSIFIER (SVC) MODEL PROVIDES A HIGHLY ACCURATE AND RELIABLE SOLUTION FOR BANKNOTE AUTHENTICATION, WITH THE CAPABILITY TO PERFORM EXCEPTIONALLY WELL IN REAL-WORLD SCENARIOS. ITS PERFORMANCE METRICS VALIDATE ITS EFFECTIVENESS AND SUITABILITY FOR DEPLOYMENT IN PRACTICAL APPLICATIONS WHERE PRECISION IN DISTINGUISHING GENUINE BANKNOTES FROM FORGED ONES IS PARAMOUNT.

THANK YOU