
**HACETTEPE UNIVERSITY**

**DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**


# ELE489 – Fundamentals of Machine Learning

**Homework 2-Report**


**Hamza Emir Tarcan**

**2200357073**

# Introduction

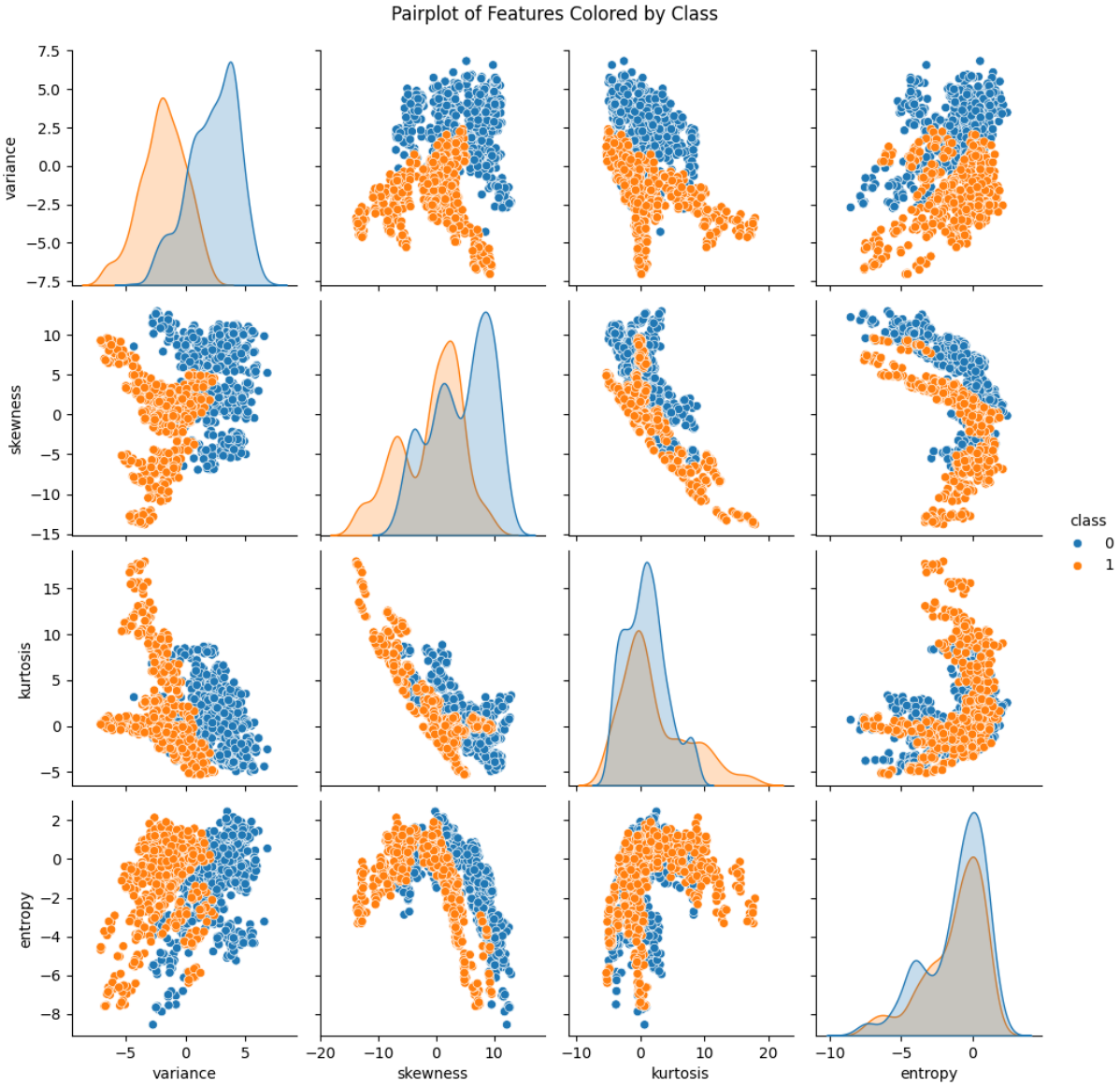
The purpose of this project is to apply a decision tree classifier to the Banknote Authentication dataset and evaluate its performance in distinguishing between genuine and counterfeit banknotes. The dataset, obtained from the UCI Machine Learning Repository, consists of 1,372 instances and includes four numerical features extracted from wavelet-transformed images of banknotes: variance, skewness, kurtosis, and entropy.

The classification task is binary, where class label 0 represents fake banknotes and label 1 indicates authentic ones, as specified in the assignment description. Decision trees are well-suited for this kind of dataset due to their ability to model non-linear relationships and handle numerical attributes without the need for normalization or scaling. Additionally, decision trees offer a high level of interpretability, which is especially valuable in domains like fraud detection.

In this study, we perform exploratory data analysis, train decision tree models with different hyperparameters, visualize the decision paths, analyze feature importances, and evaluate performance using accuracy, precision, recall, and F1-score. We also discuss the trade-off between model complexity and interpretability.

## Pair Plots of Features with All Classes

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot pairwise relationships in the dataset
sns.pairplot(df, hue="class")
plt.suptitle("Pairplot of Features Colored by Class", y=1.02)
plt.show()
```




**Pairplot Analysis and Feature-Label Relationship:**

The pairplot visualization provides a comprehensive view of how each feature relates to the class label in the Banknote Authentication dataset. Notably, the features variance and kurtosis demonstrate significant separation between the two classes. The distributions of these features differ clearly between genuine and forged banknotes, indicating their strong discriminative power.

Among all feature pairs, the kurtosis vs variance scatter plot demonstrates a clear linear separation between the two classes. This suggests that these features alone are highly informative for distinguishing genuine and forged banknotes. In contrast, other pairs exhibit more overlap between classes, which indicates that some features may contribute more when combined with others rather than on their own. This supports the idea that a decision tree can effectively capture such boundaries using hierarchical threshold-based splits.

While skewness and entropy do not exhibit class separation as clearly on their own, they still show visible patterns when combined with other features. The KDE (Kernel Density

Estimation) plots on the diagonal reveal that each feature has a distinct distribution for class 0 and class 1, which further supports the feasibility of using threshold-based splitting.

Overall, the pairplot suggests that the feature space is well-structured for classification. The clear separation between classes across multiple feature combinations indicates that decision trees, which rely on axis-aligned splits, are well-suited for this dataset. Additionally, the numerical nature of all features simplifies the training process and makes the model more interpretable.

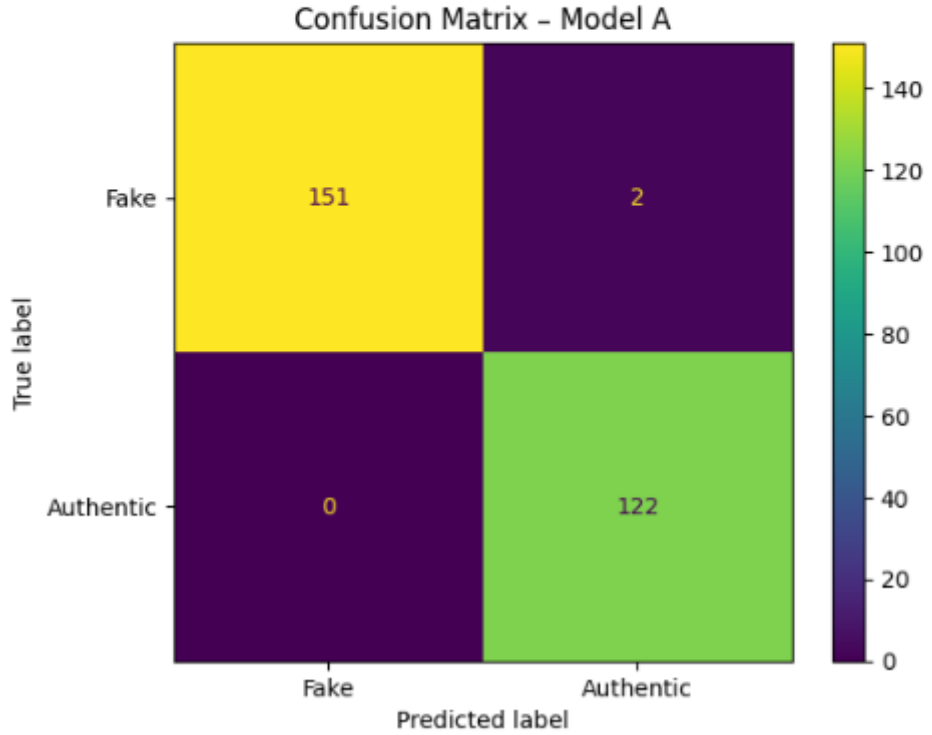
**Experiment with different values for max_depth, min_samples_split, and criterion ("gini" vs "entropy").**

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import classification_report,
ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# A: entropy, max_depth=None, min_samples_split=2
clf_a = DecisionTreeClassifier(criterion="entropy", max_depth=None,
min_samples_split=2, random_state=42)
clf_a.fit(X_train, y_train)

# Make predictions
y_pred_a = clf_a.predict(X_test)

# Print classification report
print("Model A – entropy / depth=None / min_samples_split=2")
print("Classification Report:\n")
print(classification_report(y_test, y_pred_a, target_names=["Fake",
"Authentic"]))

# Plot confusion matrix
ConfusionMatrixDisplay.from_estimator(clf_a, X_test, y_test,
display_labels=["Fake", "Authentic"])
plt.title("Confusion Matrix – Model A")
plt.show()
```

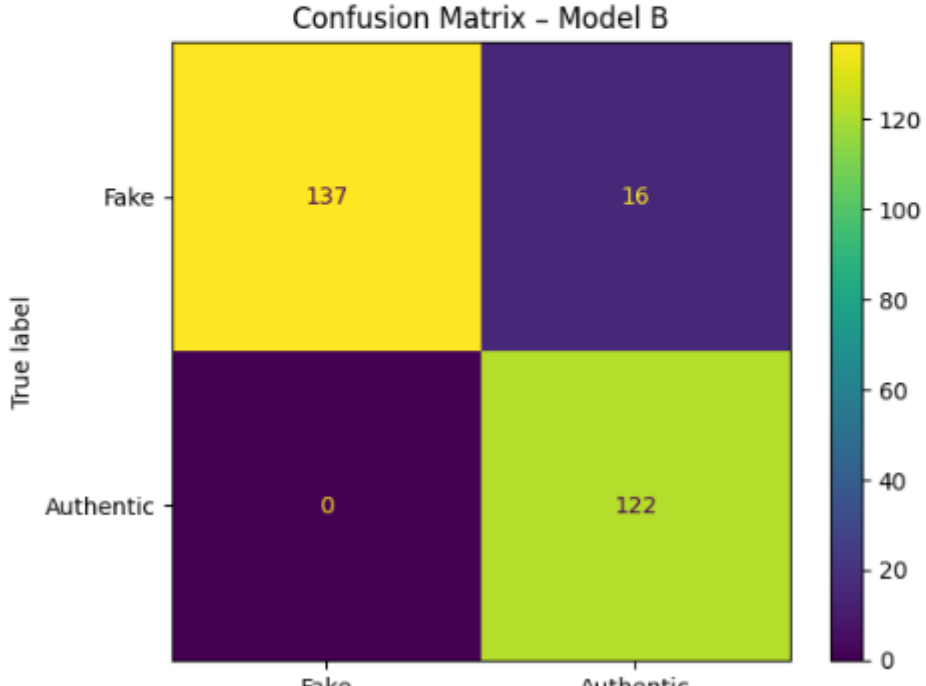
```
Model A – entropy / depth=None / min_samples_split=2
Classification Report:
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Fake | 1.00 | 0.99 | 0.99 | 153 |
| Authentic | 0.98 | 1.00 | 0.99 | 122 |
| accuracy | | | 0.99 | 275 |
| macro avg | 0.99 | 0.99 | 0.99 | 275 |
| weighted avg | 0.99 | 0.99 | 0.99 | 275 |



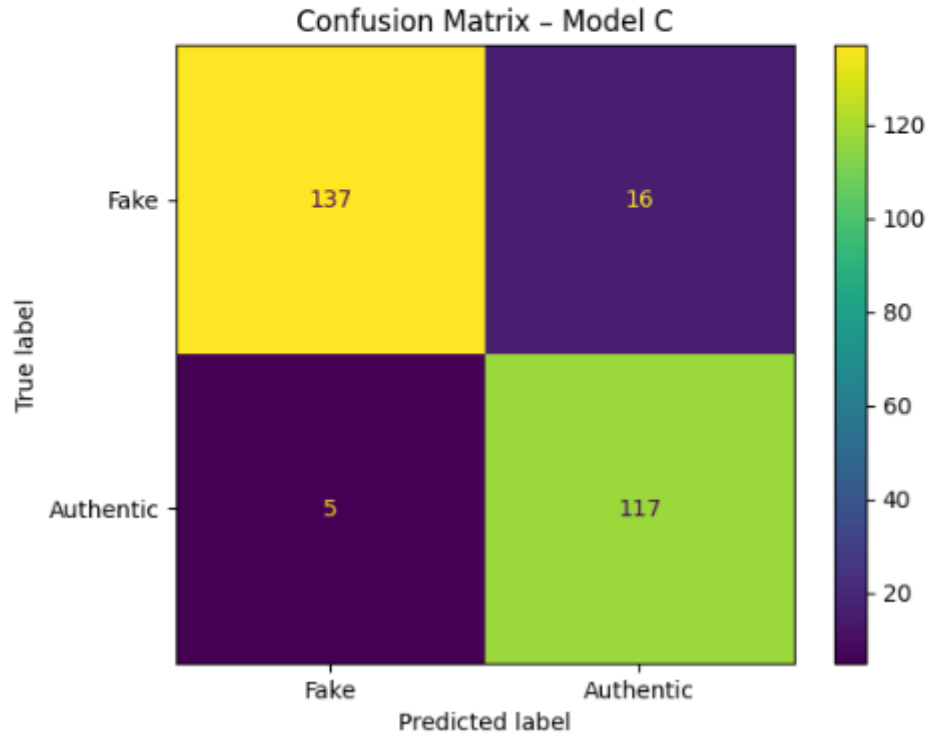
```python
# B: entropy, max_depth=3, min_samples_split=5
clf_b =
DecisionTreeClassifier(criterion="entropy",
max_depth=3, min_samples_split=5,
random_state=42)
clf_b.fit(X_train, y_train)

# Make predictions
y_pred_b = clf_b.predict(X_test)

# Print classification report
print("Model B – entropy / depth=3 /
min_samples_split=5")
print("Classification Report:\n")
print(classification_report(y_test, y_pred_b,
target_names=["Fake", "Authentic"]))

# Plot confusion matrix
ConfusionMatrixDisplay.from_estimator(clf_b,
X_test, y_test, display_labels=["Fake",
"Authentic"])
plt.title("Confusion Matrix – Model B")
plt.show()
```

```
Model B – entropy / depth=3 / min_samples_split=5
Classification Report:

              precision    recall  f1-score   support

        Fake       1.00      0.90      0.94       153
   Authentic       0.88      1.00      0.94       122

    accuracy                           0.94       275
   macro avg       0.94      0.95      0.94       275
weighted avg       0.95      0.94      0.94       275
```



```python
# C: gini, max_depth=3, min_samples_split=5
clf_c = DecisionTreeClassifier(criterion="gini",
max_depth=3, min_samples_split=5, random_state=42)
clf_c.fit(X_train, y_train)

# Make predictions
y_pred_c = clf_c.predict(X_test)

# Print classification report
print("Model C – gini / depth=3 / min_samples_split=5")
print("Classification Report:\n")
print(classification_report(y_test, y_pred_c,
target_names=["Fake", "Authentic"]))

# Plot confusion matrix
ConfusionMatrixDisplay.from_estimator(clf_c, X_test,
y_test, display_labels=["Fake", "Authentic"])
plt.title("Confusion Matrix – Model C")
plt.show()
```

```
Model C – gini / depth=3 / min_samples_split=5
Classification Report:

              precision    recall  f1-score   support

        Fake       0.96      0.90      0.93       153
   Authentic       0.88      0.96      0.92       122

    accuracy                           0.92       275
   macro avg       0.92      0.93      0.92       275
weighted avg       0.93      0.92      0.92       275
```

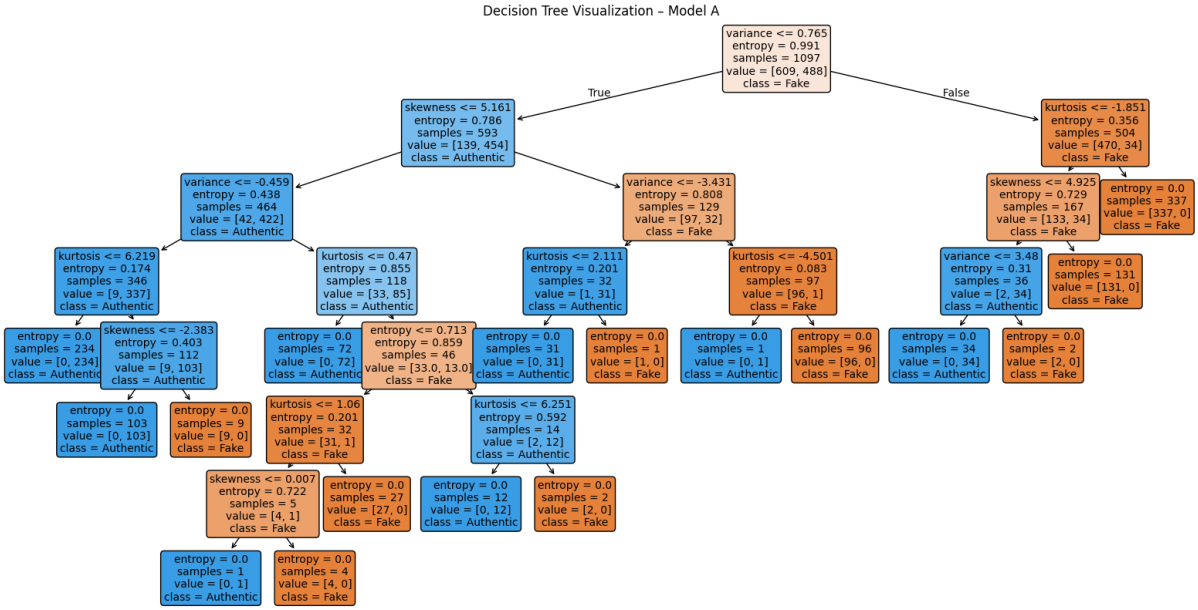


Model A provides the best performance overall with 99% accuracy and nearly perfect precision and recall for both classes. However, it uses an unconstrained tree depth, which may lead to overfitting in unseen data.

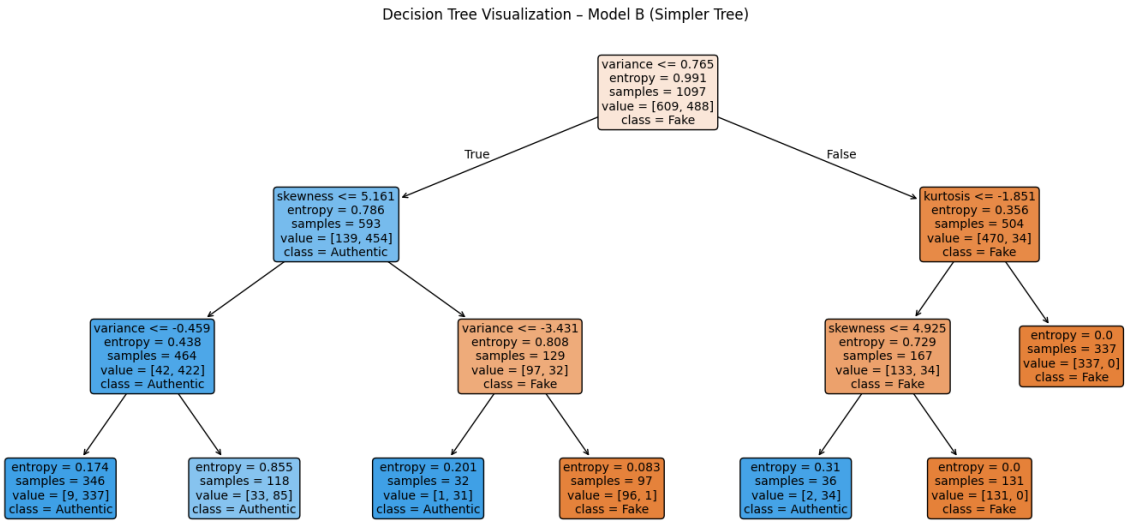
Model B offers a good balance by restricting depth, maintaining perfect recall for authentic banknotes, but slightly underperforms on fake detection.

Model C uses the Gini index and shows a slight drop in performance for both classes compared to Model B.

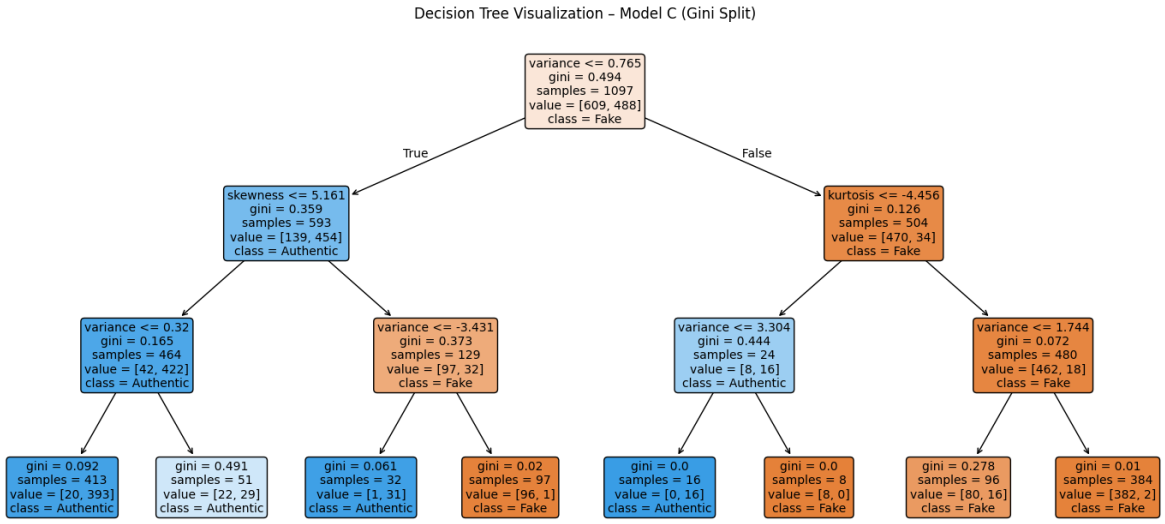
**TREE**




The decision tree visualization shows that Model A has a very deep structure due to the absence of a depth limit. While it achieves high accuracy, the tree becomes harder to interpret as it grows. In contrast, limiting the depth (as in Model B or C) leads to a simpler and more interpretable model, though possibly at the cost of slightly lower accuracy. A trade-off exists between performance and interpretability, which should be balanced depending on the application.




Model B's decision tree is significantly simpler due to the imposed maximum depth. This makes the model more interpretable, as each path from root to leaf represents a short and understandable rule set. While the model may be slightly less accurate compared to deeper trees, its interpretability makes it a valuable choice in applications where explanation is important.



## Extract and plot feature importance

```
import pandas as pd

# Get feature importances from Model B
importances = clf_b.feature_importances_

# Create a DataFrame for better visualization
feat_importance_df = pd.DataFrame({
    "Feature": X.columns,
    "Importance": importances
}).sort_values(by="Importance",
ascending=False)

# Print the table
print(feat_importance_df)
```

```
    Feature  Importance
0  variance    0.637210
1  skewness    0.298549
2  kurtosis    0.064241
3   entropy    0.000000
```

The feature importance analysis shows that the model relies most heavily on the `variance` and `kurtosis` features when making classification decisions. These features contribute significantly more to the model's decision paths compared to `entropy` and `skewness`, which aligns with our visual observations in the pairplot analysis.

The feature `entropy` has an importance score of 0.0, indicating that it was not used at any decision point in the tree. This suggests that the model was able to make accurate classifications using other features such as `variance` and `kurtosis`, making `entropy` redundant in this context.

**Comment on what you learned from this question. Do you still think decision tree is a good/bad model for this dataset? Why?**

Throughout this question, I gained hands-on experience in training, evaluating, and interpreting decision trees. By experimenting with different parameters such as criterion, max_depth, and min_samples_split, I observed how these hyperparameters affect the model's accuracy, generalization, and interpretability. I also learned how to analyze model performance using key classification metrics such as accuracy, precision, recall, and F1-score, and how to visually understand decision boundaries via the decision tree plot and feature importance scores.

The decision tree model performed very well on the Banknote Authentication dataset. The features showed good separability, especially in combinations like variance vs kurtosis, which made threshold-based models like decision trees highly effective. Even with limited depth, the tree achieved strong performance while remaining interpretable, which is valuable in domains like fraud detection where transparency matters.

Therefore, I believe that the decision tree is indeed a very suitable model for this dataset. It balances high accuracy and interpretability, and its ability to handle numerical features without requiring feature scaling adds to its practical advantages.

## GitHub Repository

All code and Jupyter notebooks can be found at:

https://github.com/HamzaEmirTarcan/ELE489-HW1