Date of publication 07/12/2023
Digital Object Identifier 10.1109/ACCESS.2023.0322000

Decision Tree Model for Banknote Verification: Classifying Authenticity with Precision

Arahanta Pokhrel[1] Amit Raj Pant[1]

Insitute of Engineering, Thapathali Campus, Final Year

Corresponding authors:
Arahanta Pokhrel (e-mail:[1]pokhrelarahanta5@gmail.com),
Amit Raj Pant (e-mail:[2]amitrajpant7@gmail.com).

ABSTRACT Genuine and forged banknotes can be distinguished based on different characteristics. These characteristics included variance, skewness, kurtosis, and entropy which can be extracted from the image of banknotes. In this work, we try to predict the authenticity of banknotes using a decision tree classifier. The dataset used contains various attributes associated with banknotes, along with corresponding classes indicating whether a banknote is genuine or forged. The decision tree model implemented utilizes entropy as a splitting criterion. A decision tree was built using some portion of the dataset and was tested on unseen data in order to evaluate the performance of the model. A confusion matrix was computed and visualized as a heatmap. Our constructed tree was able to achieve 98% accuracy on the test set of 343 images.

INDEX TERMS Authenticity. Banknote, Decision Tree, Forged, Geniune

I. INTRODUCTION

In today's digital era, where most financial transactions occur through electronic means, it is crucial to maintain the reliability and authenticity of physical currency. The significance of accurately differentiating between genuine and counterfeit banknotes cannot be overstated.In this lab, our primary objective is to analyze a robust predictive model that can effectively classify banknotes as either genuine or forged based on their unique attributes. To accomplish this, we will employ the decision tree algorithm.Decision trees are popular supervised machine learning algorithms that use a binary tree structure to assign target values to data samples. With a hierarchical graph, nodes in the tree make decisions based on features, leading to the assignment of target values in the leaves. The process of learning a decision tree involves recursively partitioning the data based on attribute values to create branches and sub-trees. Impurity measures, such as entropy, help determine the quality of splits, aiming to make the tree more homogeneous. Decision trees are effective in classifying data by creating rules at each internal node, and their human-friendly structure allows for interoperability and insights into the underlying decision-making process.We are using the potential of this decision tree algorithm and aim to create a system that automates the banknote authentication process, significantly reducing the manual effort and potential errors associated with human inspection.

Starting from the root node and making decisions at each node based on the sample's features, ultimately we reach to a target.The process of learning a decision tree involves finding optimal rules at each internal node of the tree, guided by a selected metric. The root node serves as the starting point for recursive partitioning, where the tree is divided based on attribute values. The resulting subsections of the decision tree are called branches or sub-trees, connecting the nodes of the tree. Decision nodes are formed when a sub-node splits further into additional sub-nodes, while leaf or terminal nodes are nodes without children.The fundamental idea behind decision tree algorithms revolves around selecting the best attribute or feature to split the data set. This attribute is designated as a decision node, and the data set is divided into smaller subsets accordingly.

This process is repeated recursively for each child until specific conditions are met, such as all tuples belonging to the same attribute value, no remaining attributes, or no more instances.To understand, we will focus on understanding the concept of impurity and specifically explore the use of entropy as an impurity measure in decision tree algorithms. By digging into the properties and calculations associated with entropy, we aim to gain a comprehensive understanding of how decision trees effectively partition and classify data based on attribute values.

II. METHODOLOGY

A. THEORY

A decision tree [2] is a supervised learning algorithm that employs a tree structure to make predictions or decisions based on the input. Decision trees are powerful algorithms that can be used for classification or regression predictive modeling problems. Every node in the tree represents a decision or an attribute to split the tree on. Leaf nodes are the decision nodes. We start at the root note and then process down the tree till we reach a leaf node in order to make a decision. Each node is greedy split in a such way as to maximize the information gain.

Mathematically we can express the selection process of split s is applied to a dataset D as:

$$s = \arg\max_{s \in S} IG(D, s) \tag{1}$$

where $IG(D, s)$ is the information gain when a split s is applied to a dataset D. S is the set of all possible splits.

Decision trees can be also interpreted as a set of rules to classify an input into a certain category. The backbone of decision trees is based on impurity and the goal of the decision-making process is to maximize purity as much as possible.

1) Impurity

Impurity in decision trees is the measure of uniformity of labels at a particular node. The two most widely used impurity measures are entropy and Gini Index.

The Gini Index can be expressed as:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{2}$$

where p_i is the probability of an instance in D belonging to class i.

The measure of Entropy can be mathematically formulated as:

$$\text{Entropy}(D) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{3}$$

where p_i is the probability of an instance in D belonging to class i.

2) Information Gain

The information gain is the difference between the impurity of the parent node and the weighted sum of child nodes. To measure information gain for split using entropy we can calculate it as:

$$\text{IG}(D, s) = I(D) - \frac{N_{\text{left}}}{N} \cdot I(D_{\text{left}}) - \frac{N_{\text{right}}}{N} \cdot I(D_{\text{right}}) \tag{4}$$

where:

N_{left} is instances in the left subset after the split,
N_{right} is instances in the right subset after the split,
N is the total number of instances in D,
D_{left} is the subset of D after the split on the left,
D_{right} is the subset of D after the split on the right,
$I(D)$ is the impurity measure of dataset D.

B. DATASET

The Banknote Authentication dataset [1] consists of 1327 instances of genuine and forged banknote images. Each sample in the dataset has five features extracted from the banknote images: variance, skewness, curtosis, entropy, and class. The "class" feature indicates whether a banknote is genuine or forged, with a value of 0 for forged banknotes and 1 for genuine banknotes. The first feature is the measure of variance of pixel intensities within the banknote images. It provides information about the texture and patterns present in the banknote image. It quantifies the extent or dispersion of the dataset, providing information about the distribution of pixel values.

Skewness measures the asymmetry of the pixel intensity distribution. It indicates the degree to which the distribution deviates from a symmetrical shape. Skewness assists in capturing any potential imbalances or biases present in the pixel intensities. The third feature, curtosis measures the peakedness or flatness of the distribution when compared to a normal distribution. Curtosis provides insights into the overall shape of the pixel intensity values.

The last feature, entropy, draws from information theory and captures the level of uncertainty or randomness present in the pixel intensity values of the banknote images. Higher values of entropy indicate a greater degree of unpredictability or complexity within the image.

1) Train data and test dataset

The training set is used to teach the network how to accurately perform certain tasks, the task could be a classification problem or a regression problem. In our case, we used train set to create an accurate decision tree. Whereas a test set was used to test if the tree can show similar performance on the unseen data. Evaluation of the tree on the test set checks the generalization capabilities of the model, a good performance on test set confirms the model does not only perform well on the previously seen data but can also perform accurately on new data obtained from the real world. Different parameters like accuracy, precision, and recall are calculated in order to evaluate the model on the test set.

Usually, train set is kept as large as possible, whereas a tiny portion of the dataset is used as the test set. The test set should be large enough so that it doesn't add up bias in evaluation. But making the test set too large reduces the number of examples model gets to learn from which might decrease the performance because a larger train set results in better performance. So, based on the size of dataset the split should be chosen.

C. SYSTEM BLOCK DIAGARAM

The system block diagram of the decision tree classifier is as shown in figure 1:

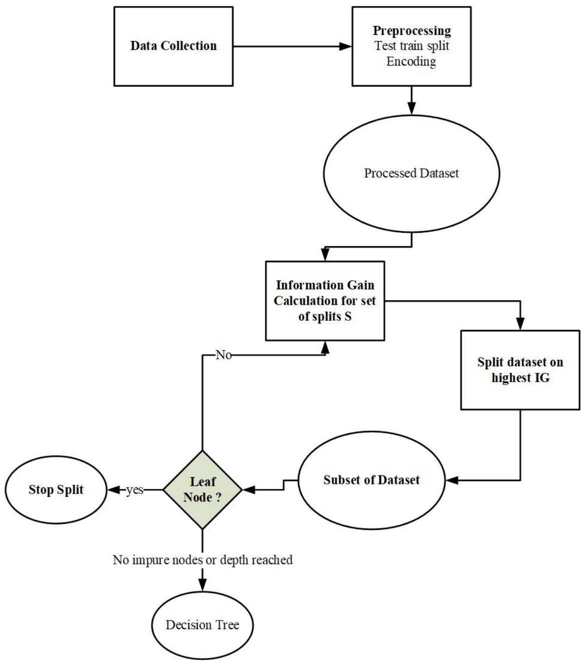

FIGURE 1: System Block Diagram

D. INSTRUMENTATION

In this lab, the following major libraries and functions were used for implementing the decision tree algorithm, performing data manipulation, and evaluating the results:

- `pandas`: This library was utilized for data manipulation and analysis. It provided functionalities for creating and manipulating data frames, allowing for easy loading and preprocessing of the dataset.
- `numpy`: The `numpy` library was used for numerical computations and operations on arrays. It provided essential mathematical functions that facilitated data manipulation and preprocessing tasks.
- `seaborn`: This library was employed for data visualization and creating informative statistical graphics. It offered a high-level interface to generate visually appealing plots, allowing for better understanding and analysis of the data.
- `plotly.express`: The `plotly.express` module was used for interactive and dynamic data visualization. It provided a range of charts and plots that could be customized and explored to gain insights from the data.
- `sklearn.tree`: The `tree` module from the `scikit-learn` library was used to access the decision tree algorithms. It provided classes and functions for building and training decision tree models.
- `train_test_split`: This function from the sklearn module was utilized to split the dataset into training and test sets. It allowed for the allocation of a certain percentage of the data for testing the trained models while keeping the remaining portion for training.
- `classification_report`: This function from the `sklearn.metrics` module was used to generate a comprehensive report of the classification results. It provided metrics such as precision, recall, F1-score, and support for each class, enabling a detailed evaluation of the

model's performance.

- `confusion_matrix`: This function from the sklearn was employed to compute the confusion matrix, which showed the number of correct and incorrect predictions made by the model. It provided valuable insights into the model's performance and potential misclassifications.
- `matplotlib.pyplot`: This library was used for data visualization, including creating plots, graphs, and figures. It provided a versatile set of functions for customizing and visualizing the decision tree and other relevant plots.

By utilizing these libraries and functions, we were able to effectively implement the decision tree algorithm, preprocess the data, split the dataset, train the models, evaluate their performance, and visualize the results in this lab.

III. WORKING PRINCIPAL

The banknote authentication dataset has four numerical attributes. To build a decision tree, the splits in the tree will be based on these attributes. The decision tree algorithm aims to find the best attribute to split the data on at each node, such that it maximizes the information gain.

To begin, data analysis is performed on the dataset to discover hidden patterns and relationships between the attributes. Various statistical and exploratory techniques may be used to gain insights into the data. This analysis helps in understanding the characteristics of the dataset and identifying which attributes are more informative for the task of banknote authentication. Once the data analysis is completed, the dataset is split into a training set and a test set in a ratio of 3:1. The training set is used to build the decision tree, while the test set is kept separate for evaluating the performance of the trained model later.

At each node of the decision tree, the algorithm calculates the information gain for splitting the dataset based on each attribute. Information gain measures the reduction in entropy or impurity achieved by splitting the data on a particular attribute. The split that results in the highest information gain is chosen as the best split for that node. After making the split, the algorithm checks the purity of the resulting child nodes. If a node is pure, it means that all the instances in that node belong to the same class. In this case, further splitting is unnecessary, and the node becomes a leaf node in the decision tree.

However, if a node is impure, which means it contains instances from different classes, the process of selecting the best attribute and splitting is repeated for that node. The algorithm continues to build the tree by recursively splitting the nodes until a certain depth is met (pre-defined stopping criterion) or until leaf nodes are obtained (nodes that are pure or meet other stopping criteria). The depth of the tree determines how many levels of splits and decision nodes are allowed. A deeper tree can capture more complex patterns in the data but runs the risk of overfitting, while a shallow tree may oversimplify the model and not capture enough information.

IV. RESULTS

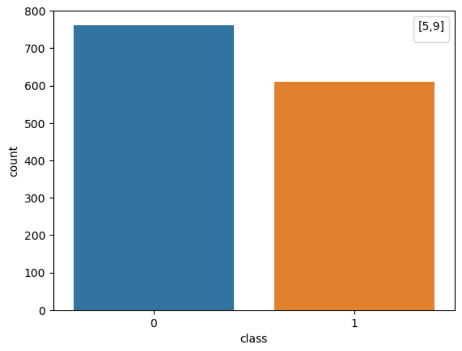

FIGURE 2: Class distribution of Banknote Authentication dataset

A. BARDIAGRAM

The bar diagram, as shown in Figure 2, visualizes the distribution of banknotes based on their authenticity. The x-axis represents the 'class' feature, which indicates whether a banknote is genuine or forged. The count of banknotes belonging to each class is represented by the height of the bars.

The diagram reveals that there are two classes in the dataset: 0 and 1. Class 0 corresponds to forged banknotes, while class 1 represents genuine banknotes. The diagram provides insight into the distribution of banknotes among these two classes. The y-axis represents the count of banknotes, indicating the number of occurrences for each class.

From the diagram, we can observe that the count for class 0 (forged banknotes) falls within the range of 700 to 800, while the count for class 1 (genuine banknotes) falls within the range of 600 to 700. This distribution allows us to gain an understanding of the relative frequency of each class within the dataset.

The x-label is set as 'class', representing the feature being displayed on the x-axis, and the y-label is set as 'count', representing the number of occurrences. The labels help to provide clear information about the variables being represented in the diagram.

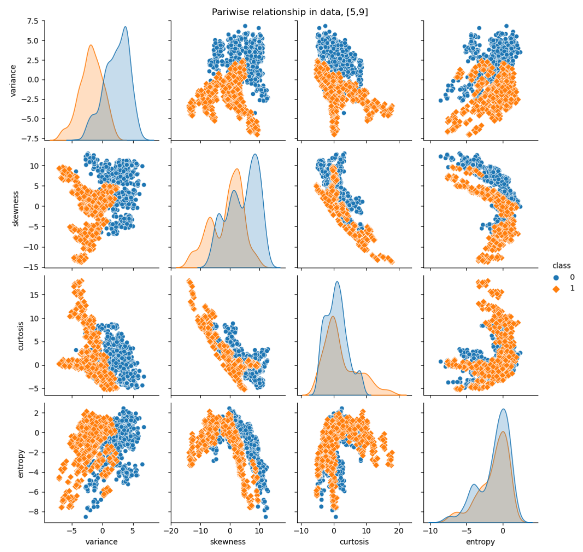

FIGURE 3: Plot of relationship between attributes

B. RELATIONSHIP BETWEEN ATTRIBUTES

We generated a pairwise plot as shown in Figure 3 using the seaborn library. This plot specifically focuses on the attributes "skewness," "curtosis," "entropy," and "class." Each scatter plot in the pairwise plot represents the relationship between two of these attributes. The x-axis and y-axis of each plot correspond to two different attributes from the dataset.

We use `markers=["o", "D"])` to visualize a pairwise plot using the seaborn library. This plot specifically focuses on the attributes "skewness," "curtosis," "entropy," and "class." Each scatter plot in the pairwise plot represents the relationship between two of these attributes. The x-axis and y-axis of each plot correspond to two different attributes from the dataset.

Understanding the relationships between these attributes is crucial for gaining insights into the dataset and informing further analysis. By visualizing the pairwise relationships, we can identify potential patterns or correlations that may exist, which can be valuable in tasks such as feature selection, understanding attribute importance, and modeling.

C. CLASSIFICATION REPORT

The table provided (Table 1) is a classification report that summarizes the performance of a classification model. Let's break down each component:

- **Precision**: Precision is a measure of how many of the predicted positive instances are actually positive. For the "Forged" class, the precision is 0.97, which means that 97% of the banknotes predicted as "Forged" are actually "Forged." Similarly, for the "Genuine" class, the precision is 0.99, indicating that 99% of the banknotes predicted as "Genuine" are actually "Genuine."
- **Recall**: Recall measures the proportion of actual positive instances that are correctly identified by the model. For the "Forged" class, the recall is 0.99, meaning that 99% of the actual "Forged" banknotes are correctly identified. For the "Genuine" class, the recall is 0.97, indicating that 97% of the actual "Genuine" banknotes are correctly identified.
- **F1-Score**: The F1-score is the harmonic mean of precision and recall. For the "Forged" class, the F1-score is 0.98, indicating a balance between precision and recall. Similarly, for the "Genuine" class, the F1-score is also 0.98.
- **Support**: Support refers to the number of instances in each class. For the "Forged" class, the support is 177, meaning there are 177 instances of "Forged" banknotes. Similarly, for the "Genuine" class, the support is 166.
- **Accuracy**: Accuracy is a measure of how well the model performs overall. In this case, the accuracy is 0.98, indicating that the model predicts the correct class for 98% of the instances.
- **Macro Avg**: Macro average calculates the average performance across all classes without considering class imbalance. The macro average precision, recall, and F1-score are all 0.98.
- **Weighted Avg**: Weighted average calculates the average performance across all classes, considering class imbalance by weighting each class based on its support. The weighted average precision, recall, and F1-score are all 0.98.

	precision	**recall**	**f1-score**	**support**
Forged	0.97	0.99	0.98	177
Genuiune	0.99	0.97	0.98	166
accuracy			0.98	343
macro avg	0.98	0.98	0.98	343
weighted avg	0.98	0.98	0.98	343

TABLE 1: Classification Report

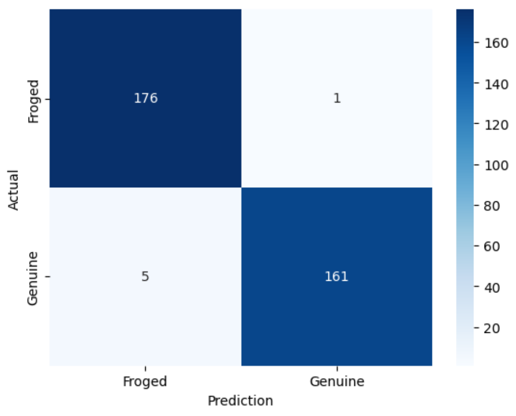

FIGURE 4: Heatmap plot of confusion matrix

D. CONFUSION MATRIX

The resulting heatmap plot shown in figure 4 visualizes the confusion matrix, providing a clear representation of the model's performance. It allows us to assess the distribution of predicted and actual values and gain insights into the accuracy and misclassifications made by the model.

The confusion matrix represents the performance of a classification model, specifically for the classes 'Forged' and 'Genuine'. The rows and columns of the matrix correspond to the predicted and actual classes, respectively. The labels for the classes are provided as 'Forged' and 'Genuine'. The matrix elements represent the counts of instances for each combination of predicted and actual classes.

Here's a breakdown of the matrix:

The top-left element (176) represents the count of instances that were correctly predicted as 'Forged' (true positives). These are the instances that were genuinely 'Forged' and were correctly classified as such by the model.

The top-right element (1) represents the count of instances that were incorrectly predicted as 'Genuine' (false positives). These are instances that were actually 'Forged' but were incorrectly classified as 'Genuine' by the model.

The bottom-left element (5) represents the count of instances that were incorrectly predicted as 'Forged' (false negatives). These are instances that were actually 'Genuine' but were incorrectly classified as 'Forged' by the model.

The bottom-right element (161) represents the count of instances that were correctly predicted as 'Genuine' (true negatives). These are the instances that were genuinely 'Genuine' and were correctly classified as such by the model.

By analyzing the values in the confusion matrix, we can gain insights into the model's performance. In this case, the model achieved a high number of true positives and true negatives, indicating its ability to correctly classify instances as either 'Forged' or 'Genuine'. However, there were a few instances that were misclassified, resulting in false positives and false negatives.

Understanding the confusion matrix allows us to evaluate the accuracy and reliability of the classification model, providing valuable information for further analysis and improvement.

E. TREE

Figure 5 presents a decision tree visualization used for classifying banknotes. The decision tree illustrates the hierarchical structure of the decision-making process and provides insights into the flow of attribute-based decisions that lead to the classification of banknotes as either genuine or forged. In the tree, the leaf nodes are distinguished by their dark blue and brown colors, while the lighter-colored nodes represent impure nodes. The dark blue nodes correspond to the class of Genuine banknotes, indicating banknotes that are verified as legitimate, while the dark brown nodes represent the class of forged banknotes, suggesting counterfeit or fraudulent banknotes.

V. DISCUSSION & ANALYSIS

From the attribute relationship plot, it can be distinguished that even two attributes variance and kurtosis are capable of separating most of the data. In addition, the plot of variance with any other attribute is capable of separating the forged and authentic banknotes to some extent. This highlights the importance of variance in separating forged bank notes from genuine notes.Also the obtained results indicate the effectiveness of our binary classification model. The high precision, recall, and F1-score values showcase the model's ability to accurately predict positive instances for both classes.

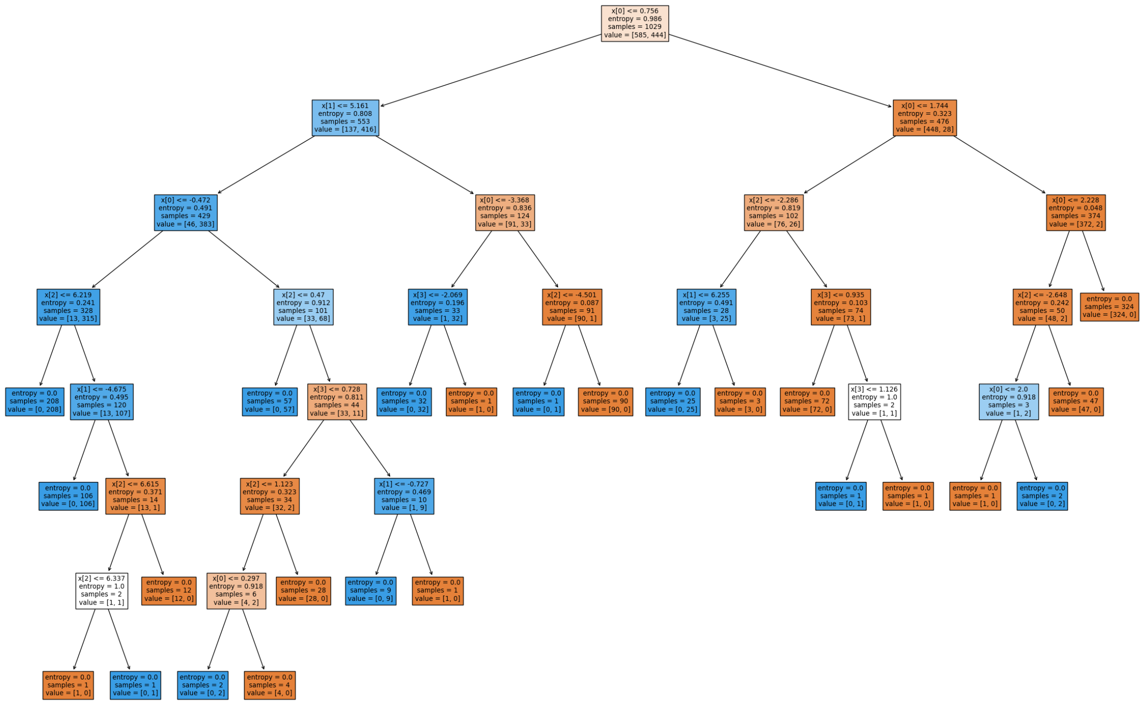
x[0] <= 0.756 entropy = 0.986 samples = 1029 value = [585, 444]
x[1] <= 5.161 entropy = 0.808 samples = 553 value = [137, 416]
x[0] <= 1.744 entropy = 0.323 samples = 476 value = [448, 28]
x[0] <= -0.472 entropy = 0.491 samples = 429 value = [46, 383]
x[0] <= -3.368 entropy = 0.836 samples = 124 value = [91, 33]
x[2] <= -2.286 entropy = 0.819 samples = 102 value = [76, 26]
x[0] <= 2.228 entropy = 0.048 samples = 374 value = [372, 2]
x[2] <= 6.219 entropy = 0.241 samples = 328 value = [13, 315]
x[2] <= 0.47 entropy = 0.912 samples = 101 value = [33, 68]
x[3] <= -2.069 entropy = 0.196 samples = 33 value = [1, 32]
x[2] <= -4.501 entropy = 0.087 samples = 91 value = [90, 1]
x[1] <= 6.255 entropy = 0.491 samples = 28 value = [3, 25]
x[3] <= 0.935 entropy = 0.103 samples = 74 value = [73, 1]
x[2] <= -2.648 entropy = 0.242 samples = 50 value = [48, 2]
entropy = 0.0 samples = 324 value = [324, 0]
entropy = 0.0 samples = 208 value = [0, 208]
x[1] <= -4.675 entropy = 0.495 samples = 120 value = [13, 107]
entropy = 0.0 samples = 57 value = [0, 57]
x[3] <= 0.728 entropy = 0.811 samples = 44 value = [33, 11]
entropy = 0.0 samples = 32 value = [0, 32]
entropy = 0.0 samples = 1 value = [1, 0]
entropy = 0.0 samples = 1 value = [0, 1]
entropy = 0.0 samples = 90 value = [90, 0]
entropy = 0.0 samples = 25 value = [0, 25]
entropy = 0.0 samples = 3 value = [3, 0]
entropy = 0.0 samples = 72 value = [72, 0]
x[3] <= 1.126 entropy = 1.0 samples = 2 value = [1, 1]
x[0] <= 2.0 entropy = 0.918 samples = 3 value = [1, 2]
entropy = 0.0 samples = 47 value = [47, 0]
entropy = 0.0 samples = 106 value = [0, 106]
x[2] <= 6.615 entropy = 0.371 samples = 14 value = [13, 1]
x[2] <= 1.123 entropy = 0.323 samples = 34 value = [32, 2]
x[1] <= -0.727 entropy = 0.469 samples = 10 value = [1, 9]
entropy = 0.0 samples = 1 value = [0, 1]
entropy = 0.0 samples = 1 value = [1, 0]
entropy = 0.0 samples = 1 value = [1, 0]
entropy = 0.0 samples = 2 value = [0, 2]
x[2] <= 6.337 entropy = 1.0 samples = 2 value = [1, 1]
entropy = 0.0 samples = 12 value = [12, 0]
x[0] <= 0.297 entropy = 0.918 samples = 6 value = [4, 2]
entropy = 0.0 samples = 28 value = [28, 0]
entropy = 0.0 samples = 9 value = [0, 9]
entropy = 0.0 samples = 1 value = [1, 0]
entropy = 0.0 samples = 1 value = [1, 0]
entropy = 0.0 samples = 1 value = [0, 1]
entropy = 0.0 samples = 2 value = [0, 2]
entropy = 0.0 samples = 4 value = [4, 0]

FIGURE 5: Decision tree of Banknote Authentication Dataset

In the tree, the decision process initiates at the root node of the tree, which is characterized by a relatively high entropy value close to one. This entropy value reflects the degree of impurity in the dataset at the root node. In other words, the dataset contains a mixture of both genuine and forged banknotes, leading to an initial lack of certainty in making accurate classifications. The root node is split based on the attribute variance, specifically using the condition x[0] <= 0.756. This splitting process generates child nodes that inherit subsets of the original dataset. By recursively repeating this splitting procedure, the decision tree branches out further, creating a hierarchy of nodes that refine the classification process. This splitting of nodes continues until we reach nodes that consist of a single class As we traverse down the tree, we can observe certain nodes that are not further split, further splitting is unnecessary as the decision tree has successfully identified subsets of the dataset that consist entirely of either genuine or forged banknotes. These nodes are referred to as leaf nodes and are characterized by their complete purity, as indicated by zero entropy. Since they contain no impurities, the leaf nodes provide definitive classifications.

VI. CONCLUSION

Through the application of decision tree algorithm, we were able to develop a classification model to identify banknotes as genuine or counterfeit with the aid of characteristics posesed by them. By implementing the decision tree classifier, we successfully achieved our objectives. The pipeline we follow in this lab includes Data importing, splitting the dataset into training and test sets, and training the decision tree models on the training data. Through the evaluation of the trained models on the test data, we were able to verify the performance predictions regarding the authenticity of banknotes. The decision tree algorithm proved to be effective in capturing the patterns and discrepancies in the banknote attributes, allowing for reliable classification. The results were obtained by utilizing entropy as the impurity measure for the decision. tree classifier. The obtained results demonstrated the effectiveness of the decision tree model in accurately predicting the authenticity of banknotes. By leveraging advanced techniques such as entropy-based splitting.The evaluation metrics, including precision, recall, F1-score, and support, provided a comprehensive assessment of the model's performance. This work provided valuable insights into the application of decision trees as binary classifiers. We successfully developed model that can classify banknotes as genuine or counterfeit with a high level of accuracy. The utilization of the decision tree algorithm, along with appropriate data preprocessing and evaluation techniques provided great learning in this lab. We learned how classification can be done using a decision tree classifier in solving real-world classification tasks.

REFERENCES

[1] Volker Lohweg. banknote authentication. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C55P57.

[2] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005.

APPENDIX A AUTHORS

ARAHANTA POKHAREL Arahanta Pokharel was born in 1999 in Biratnagar, Nepal. He is a dedicated individual with a strong passion for learning and research. Currently pursuing a Bachelor's degree in Computer Technology at the Institute of Engineering, Thapathali Campus, he is in the final year of his studies. Throughout his academic journey, he has developed a keen interest in machine learning and data science. Additionally, he has a curious mind and actively engages in quizzes and current affairs to stay updated with the latest information and is committed to acquiring new skills.

AMIT RAJ PANT Amit Raj Pant is a dedicated student pursuing his studies in the Department of Electronics and Computers at Thapathali Engineering Campus, Tribhuvan University, located in Kathmandu, Nepal. With a strong passion for technology, his interests include computer vision and machine learning on resource-constrained edge devices, which involves performing computational tasks on local devices rather than relying solely on remote servers.

APPENDIX B CODE

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
from sklearn import tree
from sklearn.model_selection import
    train_test_split
from sklearn.metrics import
    classification_report,
    confusion_matrix, roc_curve
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv('/kaggle/input/bank-
    note-authentication-uci-data/
    BankNote_Authentication.csv')

# Display the first few rows of the
    dataset
df.head()

# Display information about the
    dataset, including data types and
    missing values
df.info()

# Check for any missing values in the
    dataset
df.isnull().sum()

# Visualize the distribution of the
    target variable
ax = sns.countplot(data=df, x='class')
ax.legend(title='[5,9]')

# Visualize the pairwise relationships
     between the features, with hue
    representing the target variable
ax = sns.pairplot(df, hue="class",
    markers=["o", "D"])
ax.fig.suptitle('Pairwise relationship
     in data, [5,9]', y=1)

# Split the dataset into input
    features (X) and target variable (
    Y)
X, Y = df.iloc[:, :4], df.iloc[:, 4]

# Split the dataset into training and
    testing sets
x_train, x_test, y_train, y_test =
    train_test_split(X, Y)

# Create a decision tree classifier
    with entropy as the criterion
clf_e = tree.DecisionTreeClassifier(
    criterion="entropy")

# Train the decision tree classifier
    on the training data
clf_e.fit(x_train, y_train)

```

```python
# Make predictions on the test data
output = clf_e.predict(x_test)

# Generate a classification report to evaluate the model performance
report = classification_report(y_test, output)
print(report)

# Generate a confusion matrix to evaluate the model performance
matrix = confusion_matrix(y_test, output)
print(matrix)

# Define the class labels for the confusion matrix visualization
classes = ['Forged', 'Genuine']

# Visualize the confusion matrix
fig = sns.heatmap(matrix, cmap='Blues', fmt='g', annot=True, xticklabels=classes, yticklabels=classes)
fig.set_xlabel('Prediction')
fig.set_ylabel('Actual')

# Plot the decision tree
plt.figure()
plt.figure(figsize=(30, 20))
tree.plot_tree(clf_e, filled=True)
plt.show()
```