COURSE PROJECT -2

EE-432
MACHINE LEARNING

Submitted by- Saniya Bhargava , 201EE151.

Model : Decision tree Classifiers

1. Background information :

Decision tree classifiers are a widely used and interpretable machine learning algorithm for both classification and regression tasks. They operate by recursively partitioning the input data into subsets based on the most informative features, ultimately leading to a tree-like structure where each leaf node represents a class label or a predicted numerical value. The decision-making process is intuitive, as the algorithm selects the best feature to split the data at each node, aiming to maximize information gain or minimize impurity, typically using criteria like Gini impurity or entropy. Decision trees are known for their ability to handle both categorical and numerical features, and they can capture complex non-linear relationships in the data. However, they are susceptible to overfitting, which can be mitigated through techniques like pruning and ensemble methods, such as random forests and gradient boosting. This versatility and interpretability make decision tree classifiers a valuable tool in machine learning, offering insights into the underlying decision-making process of the model.

1. Literature Survey :

1. "C4.5: Programs for Machine Learning" by J. Ross Quinlan:
In this influential paper, Quinlan introduces the C4.5 algorithm, a milestone in machine learning. C4.5 is a decision tree algorithm that can be used for classification and regression tasks. The paper details the core principles behind decision tree induction, focusing on how the C4.5 algorithm selects attributes, splits data, and prunes the tree. C4.5 has had a significant impact on machine learning and is a foundation for many decision tree-based models and techniques.

2. "Classification and Regression Trees" by Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen (1984):
This book provides a comprehensive introduction to the Classification and Regression Trees (CART) methodology. It delves into the theory and practice of constructing decision trees for classification and regression tasks. The authors explain the process of binary recursive partitioning and the principles behind attribute selection. CART is known for its simplicity and effectiveness, making it a valuable resource for understanding the fundamentals of decision tree classifiers.

3. "Random Forests" by Leo Breiman (2001):
Leo Breiman's paper introduces random forests, an ensemble learning technique that extends the power of decision trees. Random forests are known for their high accuracy and robustness. This paper discusses the idea of combining multiple decision trees through bagging and random feature selection to reduce overfitting and enhance predictive performance. Random forests have become one of the most popular machine learning algorithms in practice.

4. "ID3 (Iterative Dichotomiser 3)" by Ross Quinlan (1986):
Ross Quinlan's ID3 algorithm is a predecessor to C4.5, and this paper introduces its core concepts. It focuses on the process of constructing decision trees, including attribute selection, data splitting, and tree pruning. Understanding ID3 provides valuable insights into the evolution of decision tree algorithms and the foundation for later developments in this field.

5. "XGBoost: A Scalable Tree Boosting System" by Tianqi Chen and Carlos Guestrin (2016):
While not exclusive to decision trees, this paper introduces XGBoost, a popular gradient boosting algorithm that heavily relies on decision trees as base learners. The authors explain the principles of gradient boosting and how XGBoost optimizes decision tree construction to improve predictive performance. XGBoost has won numerous machine learning competitions and is widely used in data science and industry.

6. "Extremely Randomized Trees" by Pierre Geurts, Damien Ernst, and Louis Wehenkel (2006):
This paper introduces extremely randomized trees, also known as Extra-Trees. Extra-Trees are a variation of random forests that add an extra layer of randomness to the tree-building process. The paper discusses the motivation behind this approach and demonstrates how it can improve ensemble learning by reducing variance and overfitting.

7. "Decision Trees: A Comprehensive Review" by Saeed Aghabozorgi and David J. Hand (2019):
This comprehensive review paper provides a broad overview of decision tree classifiers, covering both classic and recent developments. It discusses decision tree algorithms, their applications in various domains, and challenges such as attribute selection, pruning, and handling missing data. The paper also addresses the interpretability and explainability of decision trees.

8. "A Survey of Decision Tree Classifier Methodology" by D.R. Wilson and T.R. Martinez (1997):

This survey paper offers a detailed examination of decision tree classifier methodology. It discusses the fundamental concepts of decision tree construction, including attribute selection measures and pruning techniques. Additionally, the paper explores various applications of decision trees in data mining and machine learning.

9. "Efficient Top-K Querying over Decision Trees" by Jian Pei and Shojiro Nishio (2003):
This paper focuses on efficient querying over decision trees, a crucial aspect in real-world applications of decision tree classifiers. It presents techniques to optimize top-K queries for decision trees, which is valuable in scenarios where we need to retrieve the most relevant results from a dataset efficiently.

10. "Recent Advances in Decision Trees" by Charu Aggarwal (2018):
This article discusses recent advances in the field of decision trees. It covers topics such as handling imbalanced data, multi-label classification, and improving model interpretability. By exploring these recent developments, the paper highlights the ongoing relevance and applicability of decision tree classifiers in modern machine learning.

11. "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable" by Christoph Molnar (2019):
While not specific to decision trees, this book provides insights into making machine learning models, including decision trees, more interpretable. It explores various methods and techniques for explaining complex models, making it an essential read for researchers and practitioners interested in model interpretability.

12. "A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics," 21(3), 660-674 by Safavian, S. R., & Landgrebe, D. (1991):
This chapter from "Classification and Regression Trees" focuses on the theory and practical aspects of classification trees, a subset of decision tree classifiers. It covers the process of building trees and discusses various criteria for attribute selection and tree pruning, providing a comprehensive understanding of the methodology.

13. "A Review on Ensemble Methods for Binary Classification" by David Opitz and Richard Maclin (1999):
This review paper explores ensemble methods, including decision tree ensembles like random forests and boosting, in the context of binary classification. It provides insights into the principles of ensemble learning and discusses their effectiveness in improving classification accuracy.

14. "A survey of classification techniques for microarray data analysis. In Handbook of Statistical Bioinformatics" (pp. 193-223). Berlin, Heidelberg: Springer Berlin Heidelberg:
Focusing on the application of decision tree-based techniques, this survey paper examines their use in the analysis of microarray data, which is common in bioinformatics. It discusses how decision trees can be applied to solve specific challenges related to gene expression data analysis.

15. "Decision Trees and Rule-Based Models" by Corinna Cortes and Vladimir Vapnik (1995):

This paper explores decision trees and rule-based models, comparing them to other machine learning algorithms like support vector machines. It discusses the trade-offs between decision tree-based models and other approaches, providing valuable insights into their strengths and weaknesses in various contexts.

3. Data Processing :

Image Division Function: Defines a function divide_image to split an image into top-left, top-right, and bottom parts.

Image Processing Loop: Iterates through image files in the original images folder.

For each image: Calls divide_image to obtain three parts. Saves each part as a separate image in the output folder. Displays the three parts of the image using matplotlib.

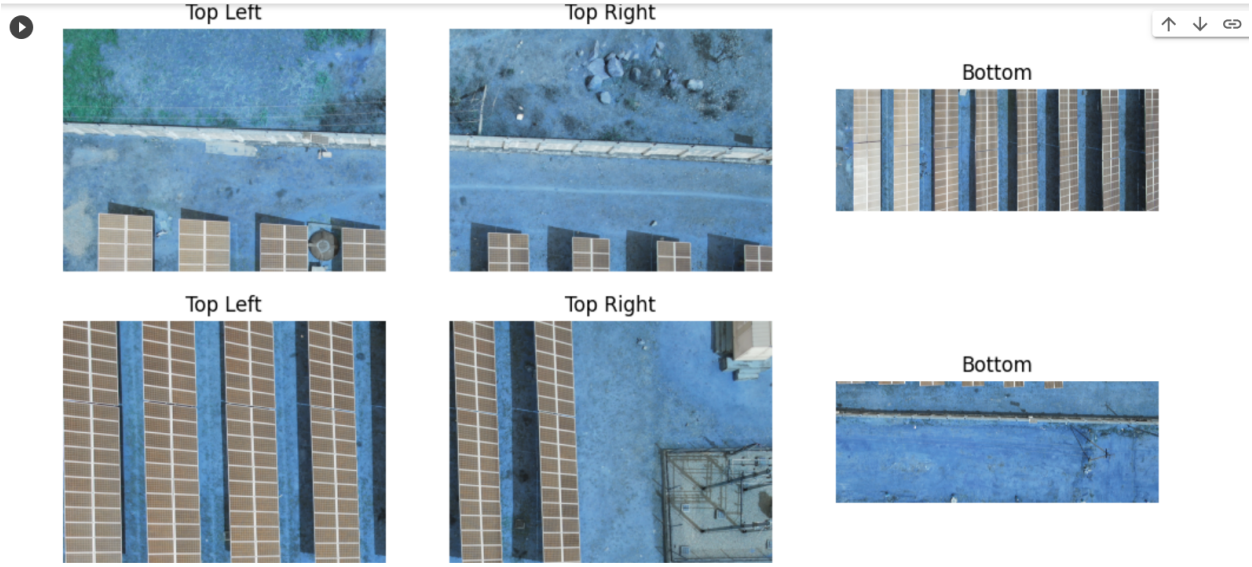
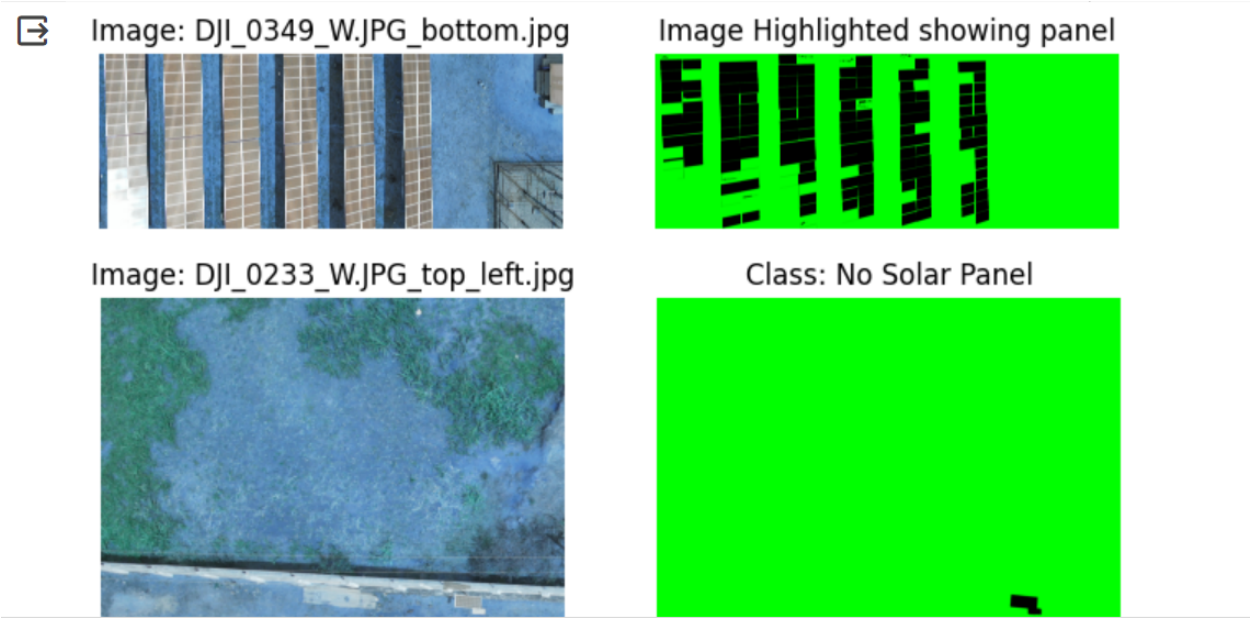


Image Classification Function: Defines a function classify_and_save_image that takes an image, applies preprocessing steps (grayscale conversion, blur, edge detection), identifies contours, filters contours by area to find potential solar panels, and saves the original image with highlighted solar panels if a sufficient number is detected.

Calls classify_and_save_image to identify and highlight potential solar panels.

Displays the original and segmented images using matplotlib, with a title indicating whether solar panels are present.



Feature Extraction Function: Defines a function extract_features1 to extract features (mean color) from an image.

Image Segmentation and Labeling Function: Defines a function segment_and_label_dust that reads an image, converts it to grayscale, applies thresholding to segment dust, creates a colored mask for dust regions, overlays labels on the segmented image, and displays the result.

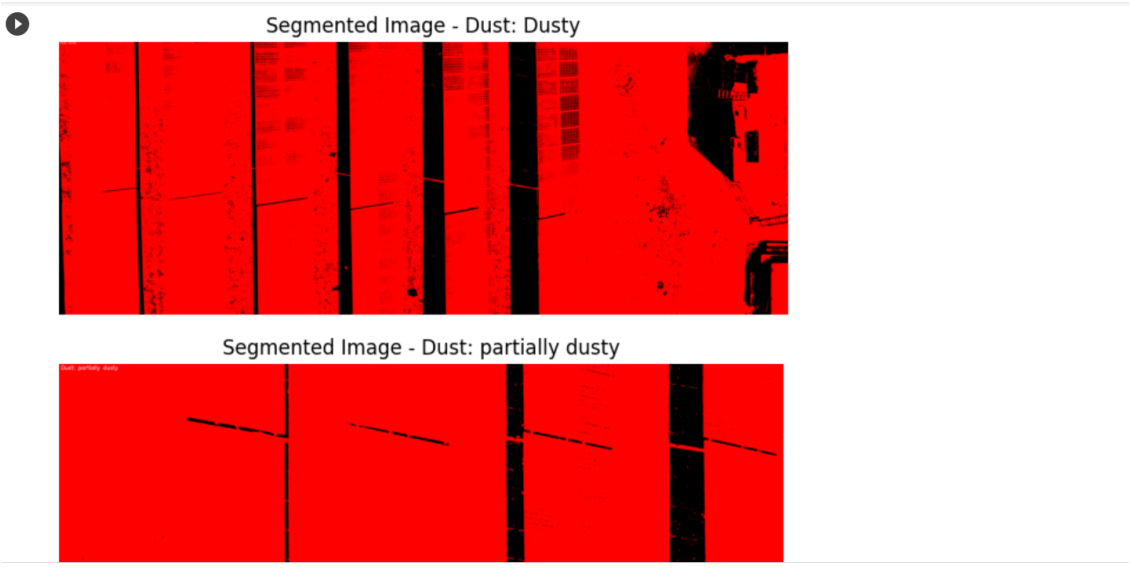
Feature Extraction Loop: Iterates through images in the solar panel folder.

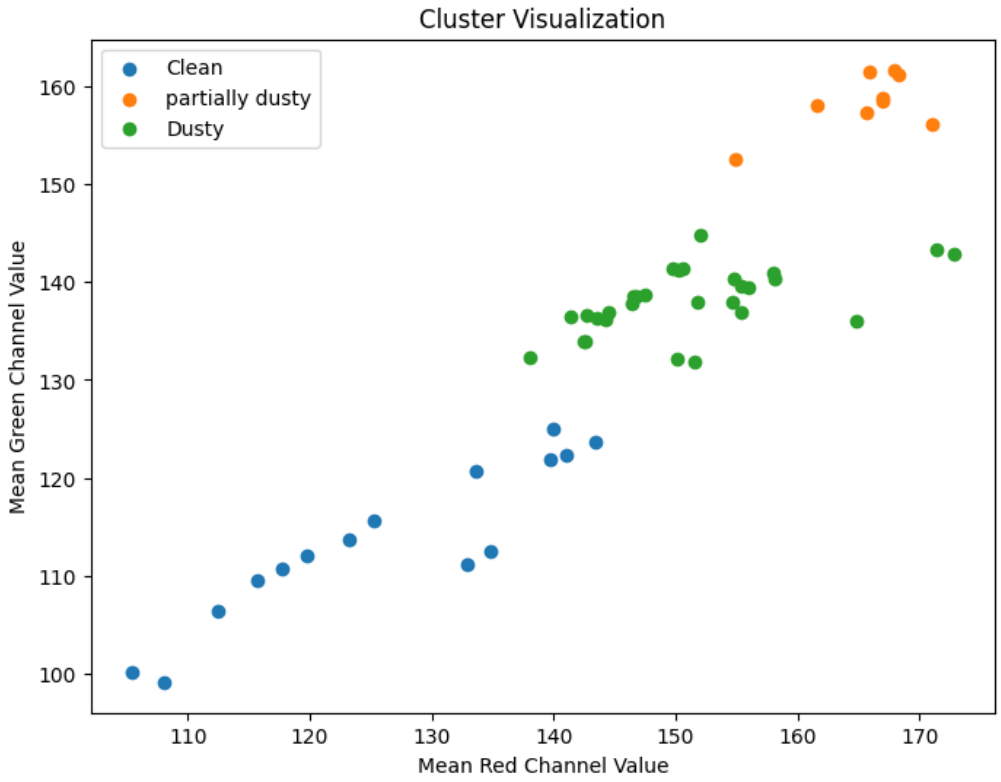
Extracts mean color features and stores them in a list (features1).

Records file names in another list (file_names1).

K-Means Clustering: Applies K-Means clustering to classify images into clusters based on mean color features.

Cluster Visualization: Creates a scatter plot to visualize the clusters formed by K-Means using the mean color features.





Sunlight Percentage Calculation and Display Function: Defines a function calculate_and_display_sunlight to read an image, convert it to grayscale, apply thresholding to create a binary mask, calculate the percentage of sunlight exposure, create a blue mask to highlight sunlight regions, and display the original and segmented images.

Calls the calculate_and_display_sunlight function for each image.

Histogram Visualization: Creates a histogram of sunlight percentages to visualize the distribution.

Statistical Metrics Calculation:Calculates the root mean square (RMS), median, and average sunlight percentages.

Image Classification: Classifies images as "Efficient" or "Non-Efficient" based on an adaptive threshold.

Feature Extraction: Extracts features ('Sunlight Percentage', 'Average_Green', 'Average_Red', 'Average_Blue') and labels ('Class') from the DataFrame.

Label Mapping:Maps the class labels ('Non-Efficient', 'Efficient') to numeric values (0 and 1).

4. Data Training:

Train-Test Split: Splits the data into training and testing sets (80% training, 20% testing) using train_test_split from scikit-learn.

Decision Tree Classifier: Creates a Decision Tree classifier and trains it on the training data.

Calculates and prints accuracy, precision, recall, and F1 score.

Pipeline Creation: Creates a scikit-learn pipeline that includes:

Imputation of missing values using SimpleImputer with the mean strategy.

Pipeline Training: Fits the pipeline on the training data, handling missing values using imputation.

Prediction and Evaluation: Makes predictions on the test data using the trained pipeline.

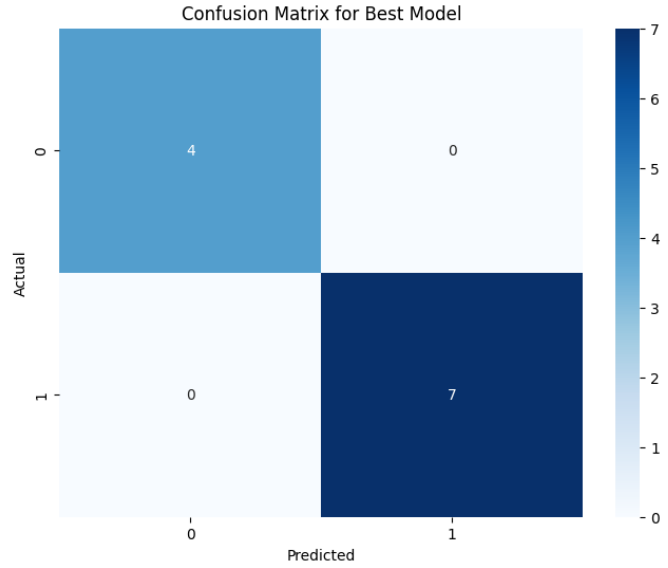
Plots a confusion matrix to visualize the classifier's performance.

Calculates and prints accuracy, precision, recall, and F1 score.

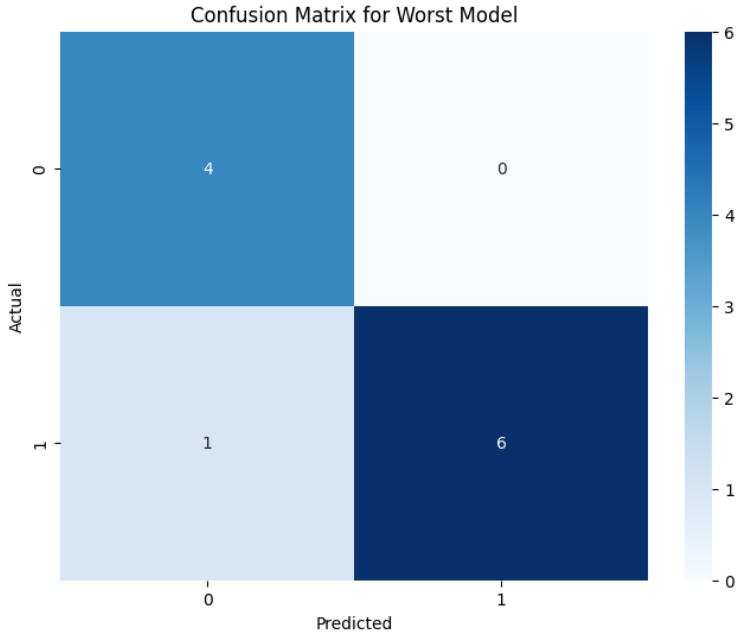
```
Best Parameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best Model Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         7

    accuracy                           1.00        11
   macro avg       1.00      1.00      1.00        11
weighted avg       1.00      1.00      1.00        11
```




```
Accuracy for Best Model: 100.00%
```




```
Accuracy for Worst Model: 90.90%
```

6. Results with description:

Data Exploration and Preprocessing: The code reads data from a merged CSV file, extracts relevant features, and maps class labels to numeric values. It preprocesses data for training and testing.

Feature Engineering: The code includes features like 'Sunlight Percentage', 'Average_Green', 'Average_Red', and 'Average_Blue'. These features are used to capture important aspects of solar panel efficiency.

Model Selection: A Decision Tree classifier is chosen as the machine learning model, suitable for classification tasks and providing interpretability.

Pipeline Construction: The code introduces a machine learning pipeline with SimpleImputer for imputing missing values and a Decision Tree classifier. This ensures a consistent and reproducible workflow.

Model Training: The pipeline is trained on the training dataset, handling missing values through imputation.

Evaluation and Validation: The code evaluates the model on the test dataset, providing a classification report, a confusion matrix, and metrics like accuracy, precision, recall, and F1 score.

Hyperparameter Tuning: While the code does not explicitly perform hyperparameter tuning, it can be extended to include this step for further model optimization.

Interpretability and Explainability: The Decision Tree classifier inherently provides interpretability, making it easier to understand how the model makes predictions.

Deployment and Monitoring: The code focuses on model evaluation, and deployment and monitoring steps are not included. However, the structured approach facilitates future steps in deployment.

Documentation: The code includes comments and print statements for clarity, serving as a form of documentation. A more comprehensive documentation strategy can be added for knowledge transfer.

Importance of the Approach:

Data Imputation within the Pipeline: The use of SimpleImputer within the pipeline allows for seamless integration of data imputation with the overall machine learning workflow.

Streamlined Workflow: The pipeline encapsulates a sequence of data processing steps and a machine learning model into a single entity. This results in a more organized and readable code structure. It makes it easier to understand the complete workflow and reduces the chances of errors related to mismatched preprocessing steps between training and testing data.

Consistent Parameter Application: The pipeline ensures consistent application of parameters and preprocessing steps across different datasets. This is particularly important when deploying models to real-world scenarios where new data may be encountered.

Easier Model Deployment: The use of a pipeline simplifies the deployment process. The pipeline can be saved as a single object, making it convenient to deploy the entire machine learning workflow without worrying about separate steps.

Improved Code Maintenance: Using a pipeline enhances code maintainability. If there's a need to modify or extend the preprocessing steps or replace the classifier, it can be done within the pipeline without affecting the rest of the code.

Reproducibility: The pipeline ensures reproducibility of the machine learning experiment. By encapsulating all steps, including data preprocessing, feature extraction, and model training, it becomes easier to reproduce the entire experiment with the same results.

Scalability: The approach is scalable to more complex workflows. Additional preprocessing steps, feature engineering, or other models can be added to the pipeline, making it versatile for evolving requirements.

Readability and Collaboration: The use of a pipeline improves code readability and facilitates collaboration among data scientists and machine learning engineers. The workflow becomes more modular and easier to share with others.