# Practical No : 2
# Galaxy classification using machine learning

Gaurav Bhoir UID :- 2309103

## Aim

The aim of this practical is to classify galaxies using simple machine learning techniques, specifically decision trees and random forests. This exercise aims to demonstrate the effectiveness of machine learning in automating galaxy classification compared to manual methods.

## Theory

### Types of Galaxies and their Classification

The classification of galaxies is primarily based on their morphology, which includes ellipticals, spirals, lenticulars, irregulars, and interacting/merging galaxies. Manual classification methods suffer from subjectivity and are time-consuming.

### Machine Learning

Machine learning enables computers to learn patterns from data without explicit programming. In this practical, we use supervised learning techniques:

- **Training and Testing:** Data is split into training and test sets. The algorithm learns patterns from the training data and is tested on unseen test data to evaluate its performance.
- **Decision Trees:** Learn simple decision rules from features to predict the target variable (galaxy type).
- **Random Forests:** Ensemble of decision trees that improve accuracy by reducing overfitting and variance.

### Confusion Matrix

A confusion matrix is used to evaluate the performance of classification algorithms. It compares the predicted classifications with actual classifications, showing true positives, true negatives, false positives, and false negatives.

## Procedure

1. Install scikit-learn library for machine learning.
2. Read the galaxy data from the provided Numpy binary file using `numpy.load()`.
3. Split the data into training and test sets.
4. Define features (u-g, g-r, r-i, i-z, eccentricity, adaptive moments, Petrosian flux) and targets (galaxy types).
5. Train a decision tree classifier on the training set and predict on the test set.
6. Evaluate accuracy using a confusion matrix.
7. Repeat the process using a random forest classifier and plot the confusion matrix.

## Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Step 1: Load the data using pandas
data = pd.DataFrame(np.load('/content/galaxies.npy'))

# Step 2: Split the data into features (X) and target (y)
X = data.iloc[:, :-1]  # Features: all columns except the last one
y = data.iloc[:, -1]   # Target: last column

h = [0.2, 0.5, 0.9]
for test_size in h:
    # Step 3: Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
        random_state=42)

    # Step 4: Set up a decision tree classifier
    dt_classifier = DecisionTreeClassifier(random_state=42)

    # Step 5: Train the decision tree classifier
    dt_classifier.fit(X_train, y_train)

    # Step 6: Apply the classifier to the test data
    y_pred = dt_classifier.predict(X_test)

    # Step 7: Evaluate accuracy using confusion matrix
    accuracy_dt = accuracy_score(y_test, y_pred)
    print(f"Decision Tree Accuracy with test size {test_size}: {accuracy_dt:.2f}")

    # Plot confusion matrix for decision tree
    cm_dt = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
    # Print confusion matrix for decision tree
    print(f"Confusion Matrix - Decision Tree Classifier with test size
        {test_size}:\n{cm_dt}\n")

    # Step 8: Set up a random forest classifier
    rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

    # Step 9: Train the random forest classifier
    rf_classifier.fit(X_train, y_train)

    # Step 10: Apply the classifier to the test data
    y_pred_rf = rf_classifier.predict(X_test)

    # Step 11: Evaluate accuracy using confusion matrix
    accuracy_rf = accuracy_score(y_test, y_pred_rf)
    print(f"Random Forest Accuracy with test size {test_size}: {accuracy_rf:.2f}")

    # Plot confusion matrix for random forest
    cm_rf = confusion_matrix(y_test, y_pred_rf)
    ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
    # Print confusion matrix for random forest
    print(f"Confusion Matrix - Random Forest Classifier with test size
        {test_size}:\n{cm_rf}\n")
```

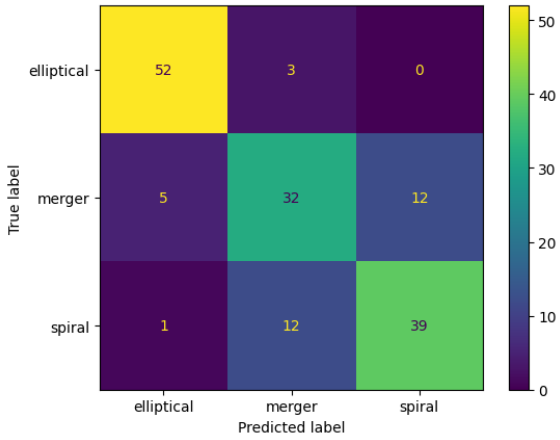


Figure 1: Decision Tree Classifier

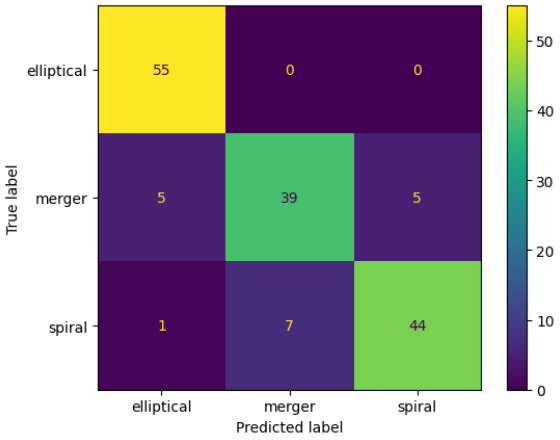


Figure 2: Random Forest Classifier

Figure 3: *
Test Size = 0.2

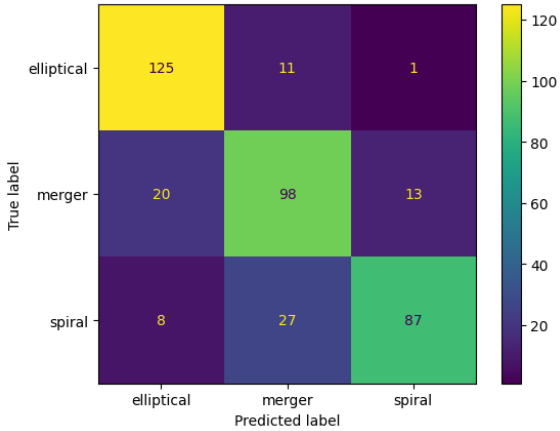


Figure 4: Decision Tree Classifier

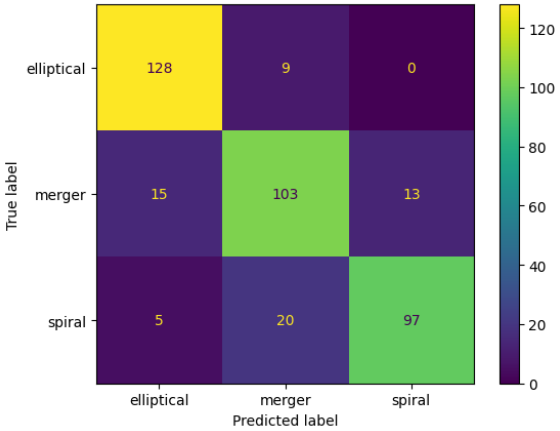


Figure 5: Random Forest Classifier

Figure 6: *
Test Size = 0.5

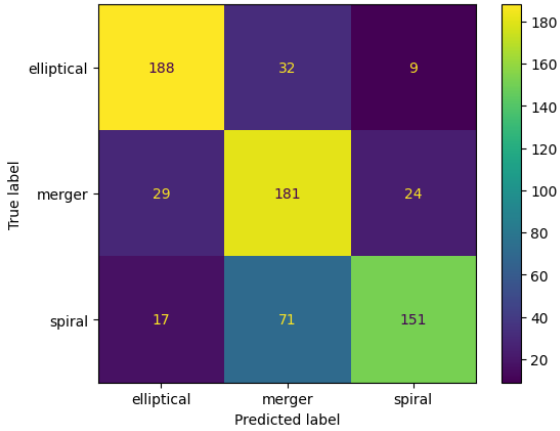


Figure 7: Decision Tree Classifier

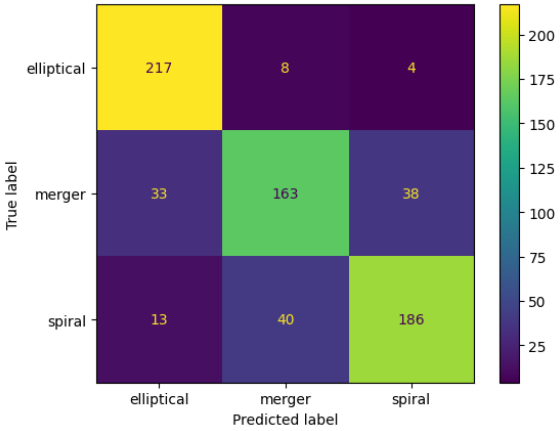


Figure 8: Random Forest Classifier

Figure 9: *
Test Size = 0.9

# Conclusion

In this practical, we evaluated the performance of Decision Tree and Random Forest classifiers for galaxy classification using different test sizes. The key observations from our experiments are summarized as follows:

## Impact of Increasing Test Size

As the test size increases, the proportion of data used for testing grows, while the proportion of data used for training decreases. This shift impacts the performance of our classifiers in the following ways:

- **Model Accuracy:** We observed a slight decrease in accuracy for both Decision Tree and Random Forest classifiers as the test size increased. This is expected because a smaller training set provides less information for the model to learn from, which can lead to a less robust model.
- **Confusion Matrix:** The number of correctly and incorrectly classified instances in the confusion matrix fluctuates with test size. With a larger test size, we may see more misclassifications as the model has less training data to generalize from.

## Impact of Decreasing Test Size

Conversely, decreasing the test size means more data is available for training the model, while less data is used for testing. This impacts the performance in the following ways:

- **Model Accuracy:** Generally, a smaller test size can lead to higher accuracy as the model has more training data to learn from, which can result in better generalization. However, this also depends on the nature of the data and the complexity of the model.
- **Confusion Matrix:** With more data available for training, the model tends to perform better on the test set, leading to fewer misclassifications. However, a very small test size might not provide a comprehensive evaluation of the model's performance.

## General Observations

- **Decision Tree Classifier:** The Decision Tree classifier showed a noticeable decrease in accuracy with an increase in test size. This is likely due to its high variance nature, where it can overfit the training data and perform poorly on unseen data.
- **Random Forest Classifier:** The Random Forest classifier, being an ensemble method, generally showed more stable performance across different test sizes. The ensemble nature helps in reducing overfitting, thus providing more consistent results.

In summary, the test size plays a critical role in evaluating the performance of machine learning models. A balanced approach is necessary to ensure that enough data is available for both training and testing to achieve a robust and reliable model evaluation.