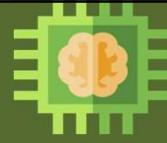


**Elective Course**

Course Code: CS4103

Autumn 2025-26

**Lecture #37**

# Artificial Intelligence for Data Science

**Week-11:****MACHINE LEARNING (Part V)****Exploring Decision Tree using Python****Course Instructor:**

Dr. Monidipa Das

Assistant Professor

Department of Computational and Data Sciences

Indian Institute of Science Education and Research Kolkata, India 741246

## DecisionTreeClassifier from Scikit-learn



```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini',
splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=None, random_state=None,
max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None,
ccp_alpha=0.0, monotonic_cst=None)
```

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

- Although decision trees are theoretically capable of handling categorical data, the `sklearn.tree.DecisionTreeClassifier` requires numerical representation of all features;
- perform an appropriate encoding step as part of data preprocessing;

## Decision Tree for Iris Classification: Dataset Loading



```

import pandas as pd
#Load the dataset
dataset =
pd.read_csv('F:/CS4103/code/iris.csv')
#View dataset
print("First 6 rows of the dataset:")
print(dataset.head(6))

#Dataset column names
print("\nName of the columns in the dataset:")
print(list(dataset.columns))

input_features = list(dataset.columns[:-1])
class_col=dataset.columns[-1]
print("Name of the input feature columns:",input_features)
print("Name of the class/label column:",class_col)

```

First 6 rows of the dataset:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
5	5.4	3.9	1.7	0.4	Setosa

Name of the columns in the dataset:  
['sepal.length', 'sepal.width',  
'petal.length', 'petal.width', 'variety']

Name of the input feature columns:  
['sepal.length', 'sepal.width',  
'petal.length', 'petal.width']

Name of the class/label column: variety

Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree for Iris Classification: Exploring Dataset Details



```

# Number of rows or instances and number of columns features
print("\nTotal number of data points/examples/instances:", dataset.shape[0])
print("Total number of input features:", dataset.shape[1]-1)

#Dataset description
print("\nDescription of the dataset:")
print(dataset.describe())

#Print number of classes
c=dataset[class_col].nunique()
print("\nNumber of classes(discrete labels):",c)

#Number of instances per class
print("\nSample count per class:")
print(dataset[class_col].value_counts())

#Check missing values in variables
dataset.isnull().sum()

```

Total number of data points/examples/instances: 150  
Total number of input features: 4

Description of the dataset:

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Number of classes (discrete labels): 3

Sample count per class:

	Setosa	Versicolor	Virginica
name	50	50	50

Name: variety, dtype: int64

Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree for Iris Classification: Data Split



```
X = dataset[input_features].values
y = dataset[class_col].values

#split dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

#check the shape of X_train and X_test
print("Traning set shape:",X_train.shape, "\nTest set shape:",X_test.shape)
```

Traning set shape: (120, 4)  
Test set shape: (30, 4)

Dr. Monidipa Das, Department of CDS, IISER Kolkata

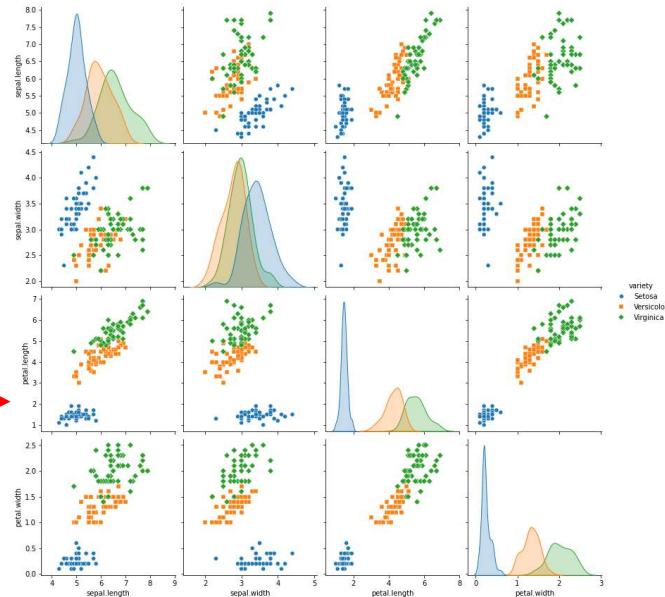
## Decision Tree for Iris Classification: Data Visualization



```
#visualize the distribution
of features and their
relationship with others

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure()
sns.pairplot(dataset, hue =
class_col, size=3, markers=["o",
"s", "D"])
plt.show()
```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree for Iris Classification: Building Classifier



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

#instantiate the DecisionTreeClassifier model with criterion entropy
clf_ent= DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=0)

#fit the model
clf_ent.fit(X_train, y_train)
y_pred_ent = clf_ent.predict(X_test)
```

Measures of impurity

$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk}) \quad p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

#Evaluating predictions

```
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred_ent)
print(cm)
```

$$\begin{bmatrix} [11 & 0 & 0] \\ [0 & 13 & 0] \\ [0 & 0 & 6] \end{bmatrix}$$

Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree for Iris Classification: Evaluating Classifier



```
#Evaluating predictions
disp = ConfusionMatrixDisplay(confusion_matrix=
cm,display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
```

#print the scores on training and test set

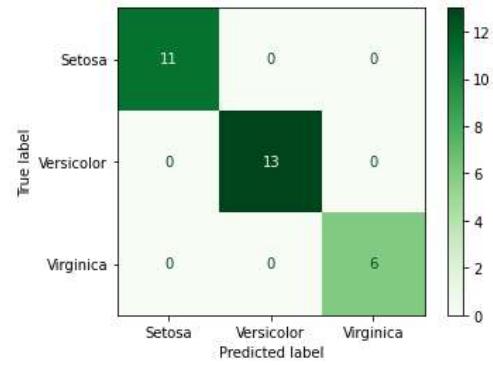
```
print('Training set score:\n{:.4f}'.format(clf_ent.score(X_train, y_train)))
print('Test set score:\n{:.4f}'.format(clf_ent.score(X_test, y_test)))
```

from sklearn.metrics import classification\_report

#Comprehensive classification report

```
report = classification_report(y_test, y_pred_ent)
print("\nClassification Report:\n", report)
```

Training set score: 0.9917  
Test set score: 1.0000



	precision	recall	f1-score	support
Setosa	1.00	1.00	1.00	11
Versicolor	1.00	1.00	1.00	13
Virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Dr. Monidipa Das, Department of CDS, IISER Kolkata

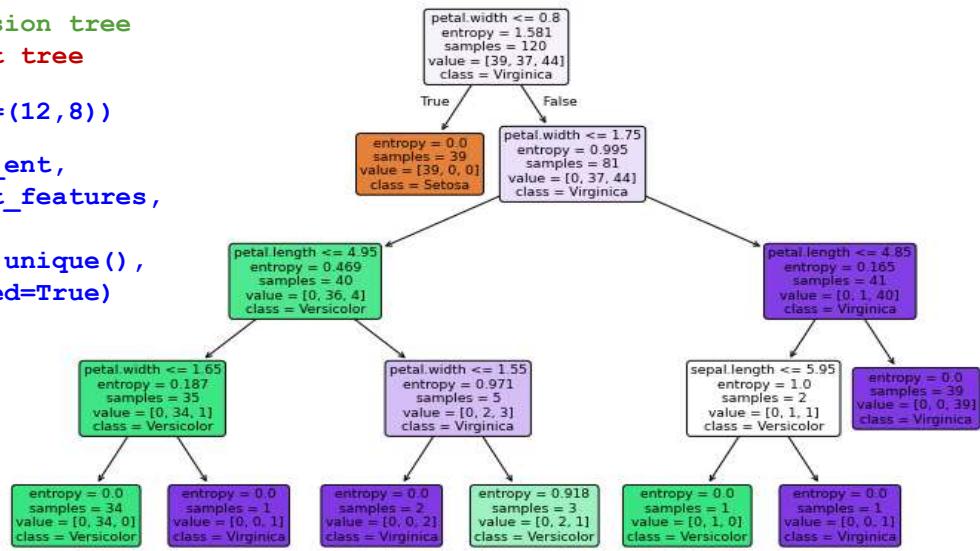
## Decision Tree for Iris Classification: Tree Plotting



```
#plotting the decision tree
from sklearn import tree

plt.figure(figsize=(12,8))

tree.plot_tree(clf_ent,
feature_names=input_features,
class_names=
dataset[class_col].unique(),
filled=True, rounded=True)
```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree for Iris Classification: Building Classifier (2)



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

#instantiate the DecisionTreeClassifier model with criterion gini index
clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=0)

#fit the model
clf_gini.fit(X_train, y_train)
y_pred_gini = clf_gini.predict(X_test)

#Evaluating predictions
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred_gini)
print(cm)
```

### Measures of impurity

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk}) \quad p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

$\begin{bmatrix} [11 & 0 & 0] \\ [0 & 13 & 0] \\ [0 & 0 & 6] \end{bmatrix}$
---

Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree for Iris Classification: Evaluating Classifier (2)

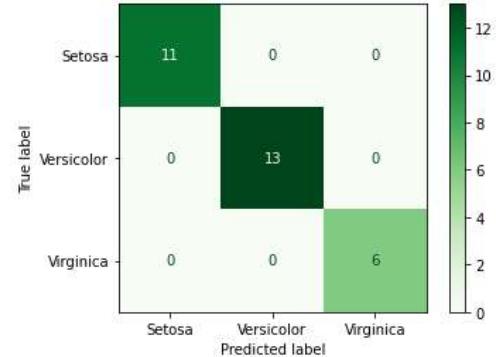
11

```
#Evaluating predictions
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

#print the scores on training and test set
print('Training set score:\n{:.4f}'.format(clf_gini.score(X_train, y_train)))
print('Test set score:\n{:.4f}'.format(clf_gini.score(X_test, y_test)))

from sklearn.metrics import classification_report
#Comprehensive classification report
report = classification_report(y_test, y_pred_gini)
print("\nClassification Report:\n", report)

Training set score: 0.9917
Test set score: 1.0000
```



Classification Report:				
	precision	recall	f1-score	support
Setosa	1.00	1.00	1.00	11
Versicolor	1.00	1.00	1.00	13
Virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Dr. Monidipa Das, Department of CDS, IISER Kolkata

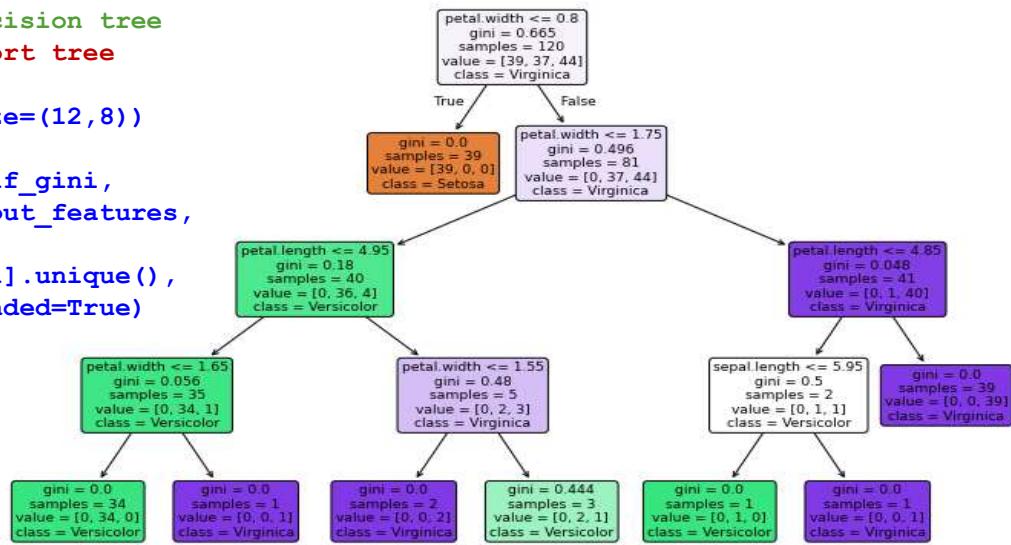
## Decision Tree for Iris Classification: Tree Plotting (2)

12

```
#plotting the decision tree
from sklearn import tree

plt.figure(figsize=(12,8))

tree.plot_tree(clf_gini,
feature_names=input_features,
class_names=
dataset[class_col].unique(),
filled=True, rounded=True)
```



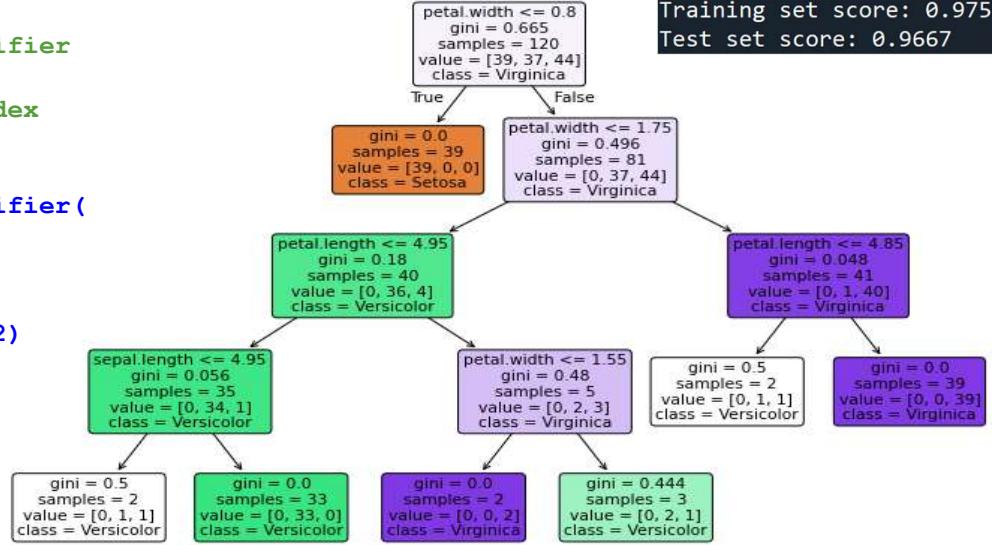
Dr. Monidipa Das, Department of CDS, IISER Kolkata

# DTC with Changed Hyperparameter



```
# instantiate the
DecisionTreeClassifier
(DTC) model with
criterion gini index

clf_gini =
DecisionTreeClassifier(
criterion='gini',
max_depth=4,
random_state=0,
min_samples_leaf=2)
```



Training set score: 0.9750  
Test set score: 0.9667

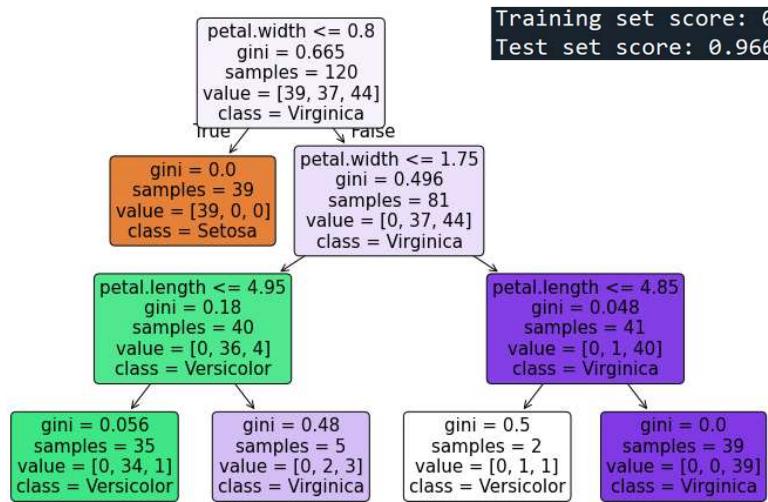
Dr. Monidipa Das, Department of CDS, IISER Kolkata

# DTC with Changed Hyperparameter (2)



```
# instantiate the
DecisionTreeClassifier
(DTC) model with
criterion gini index

clf_gini =
DecisionTreeClassifier(c
riterion='gini',
max_depth=3,
random_state=0)
```



Training set score: 0.9667  
Test set score: 0.9667

Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree Classifier: Decision Boundary



```

import matplotlib.pyplot as plt
import numpy as np
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.tree import DecisionTreeClassifier

feature_1, feature_2 = np.meshgrid(
    np.linspace(X[:, 0].min(), X[:, 0].max()),
    np.linspace(X[:, 1].min(), X[:, 1].max()))
)

grid = np.vstack([feature_1.ravel(), feature_2.ravel()]).T

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train1 = le.fit_transform(y_train)
y1 = le.fit_transform(y)

```

Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree Classifier: Decision Boundary



```

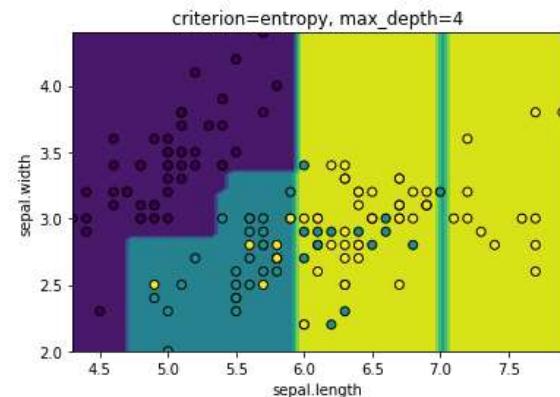
tree = DecisionTreeClassifier(criterion='entropy', max_depth=4,
random_state=0).fit(X_train[:, :2], y_train1)

y_pred = np.reshape(tree.predict(grid), feature_1.shape)

display =
DecisionBoundaryDisplay(xx0=feature_1,
xx1=feature_2, response=y_pred)
display.plot()
display.ax_.scatter(
    X[:, 0], X[:, 1], c=y1, edgecolor="black")

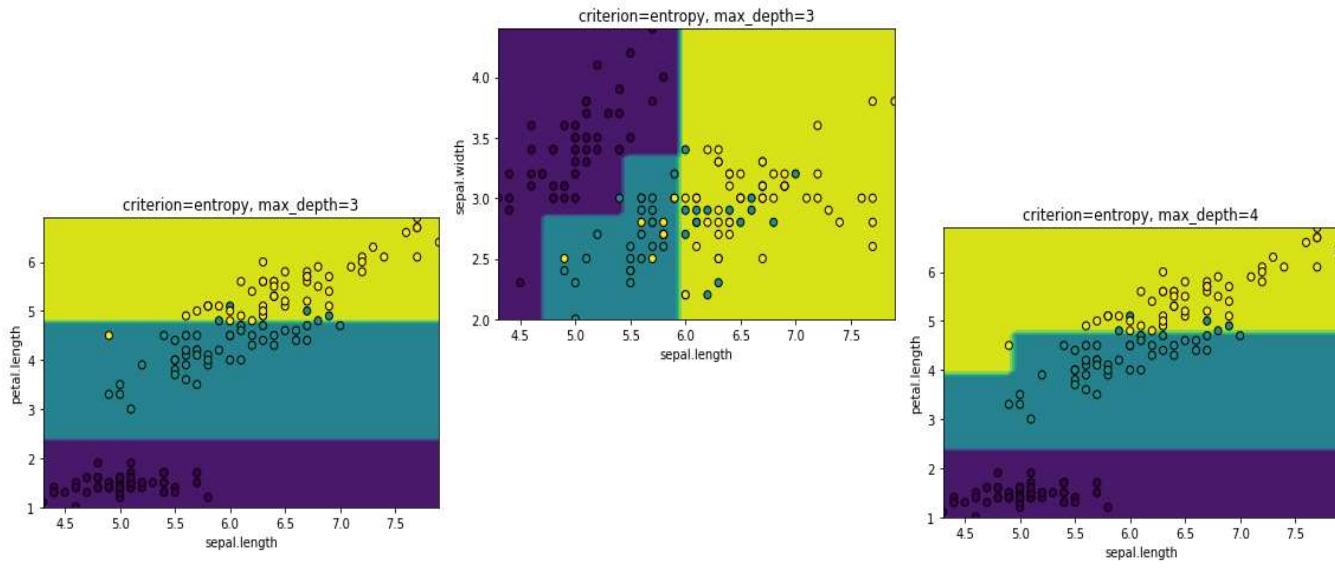
plt.xlabel(input_features[0])
plt.ylabel(input_features[1])
plt.title('criterion=entropy, max_depth=4')
plt.show()

```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Decision Tree Classifier: Decision Boundary



Dr. Monidipa Das, Department of CDS, IISER Kolkata

## Home Assignment



[https://drive.google.com/file/d/1yExdlxkm6VjL2jEH\\_IgAKYC1Aj9k392E/view?usp=sharing](https://drive.google.com/file/d/1yExdlxkm6VjL2jEH_IgAKYC1Aj9k392E/view?usp=sharing)

- Above is the link for the full **Steel-Plate-Fault-Detection Dataset** comprising of samples having **7 different types of faults**:
  - Pastry, Z\_Scratch, K\_Scratch, Stains, Dirtiness, Bumps, Other\_Faults
- Each sample has **27 Input Features**:
  - X\_Minimum, X\_Maximum, Y\_Minimum, Y\_Maximum, Pixels, Areas\_X, Perimeter\_Y, Perimeter, Sum\_of\_Luminosity, Minimum\_of\_Luminosity, Maximum\_of\_Luminosity, Length\_of\_Conveyer, TypeOfSteel\_A300, TypeOfSteel\_A400, Steel\_Plate\_Thickness, Edges\_Index, Empty\_Index, Square\_Index, Outside\_X\_Index, Edges\_X\_Index, Edges\_Y\_Index, Outside\_Global\_Index, LogOfAreas, Log\_X\_Index, Log\_Y\_Index, Orientation\_Index, Luminosity\_Index, SigmoidOfAreas
- Assume *Train-Test split ratio* to be 80:20, and *random\_state = 0*
- Develop **KNN and Decision Tree Classifiers**. Use **appropriate data preprocessing, if required**.
- Show **comparative performance study** considering both the models

Dr. Monidipa Das, Department of CDS, IISER Kolkata

# Home Assignment [contd.]



- Submission Deadline: **15-NOV-2025**
- To be done by a group of **maximum four (04)** students;
- Submit through email with subject "**CS4103: Assignment on Building ML Classifier for Steel-Plate Fault Detection**" and cc to your group-mates, if there is any
- **Please write the code yourself. DO NOT use AI tools to develop the code.**

Dr. Monidipa Das, Department of CDS, IISER Kolkata



# Questions?

Dr. Monidipa Das, Department of CDS, IISER Kolkata