13 Oct

Day 8

SCIKIT LEARN

Goal:

AI → is to make systems intelligent | Siri
ML → we teach machines from data | Learning patterns | Predict HousePrices
DL → Uses neural networks to automatically learn complex patterns | GPT, self-driving

ML → Scikit learn
DL → Pytorch

↳ A python library for classical machine learning. It provides tools for data preprocessing (1), training, evaluation, model selection (2) using algorithms like regression, classification etc

→ used for smaller datasets, structured data

→ Sklearn does not have neural networks like CNN, RNN, Transformer

It provides ML Algos (mathematically defined)

- Regression → Linear
- Classification → logistic Regression, Random Forest, SVM, Decision Tree
- Clustering → KMeans
- Dimensionality Reduction → PCA
- Preprocessing → Scaling

Convention "X" → Input / Features (multiple cols)

[[200, 3, 8-2], [4, 3, 2], ...] → Matrix

"y" → Output / Target (Vector = a single col'n)

Ex: $y = WX + b$ In Linear Regression

⇒ X_train, X_test, y_train, y_test = train_test_split (X, y, test_split = 0.2)

(CANNOT CHANGE ORDER)

- Randomly select 80% rows (put features & labels in X_train & y_train)
- Remaining 20% rows, put their features → X_test, labels → y_test

∴ Each 'X' row stays with its correct 'y' value
∴ Model learns from X_train, y_train
AND
model's accuracy is checked using X_test, y_test

Overview
↳ Ex3 Classification

1. from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import Train_Test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import kNeighboursClassifier.

2. X, y = load_breast_cancer(return_X_y = True)
X → multidimensional numpy array
y → 0, 1

3. X_train, ... = train_test_split (X, y, test_size = 2)

4. scaler = StandardScaler()
X_train_scaled = scaler.fit_transform (X_train)
X_test_scaled = scaler.transform (X_test)

fit_transform
↳ calculates (min, max) means, std | learns parameters to scale data
↳ Applies scaling

→ X_test : we don't want to do (fit_transform), as that (20% data) is for testing/predicting purposes, not training
we don't want to calculate its mean, std again, as it will be misleading good.
Test data is scaled on unseen data / as if it is new
no sense in (training & test data)?

(5) knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)

(6) print(knn.score(X_test_scaled, y_test))
O/p → .95 = 95% accuracy
correct classification

WORKFLOW

(1) We have a dataset → split, scale
(2) We either want to do Regression, Classification, Clustering.
(3) We then train on that data
(4) We want to make predictions with our trained model.
(How well our model will perform on unseen data.)

→ load, fetch, make. from sklearn.datasets import