Lab 4

Implement Random Forest ensemble method on a given dataset.

- Input:
  - Load the dataset D with n samples and m features.
  - Number of trees to grow T
  - Number of features to consider at each split F
  - Maximum tree depth MaxDepth
  - Minimum samples required to split a node minSampleSplit

- Step 1: Initialize
  Create an empty list forest = [] to store decision trees.

- Step 2: Build T trees

  for t = 1 till T:
  - Bootstrap Sampling:
    Create dataset D_t by sampling n examples with replacement from D.
  - Build a decision tree using D_t
    - At each node of the tree randomly pick F features from all available features
    - For each feature:
      - try different split thresholds
      - calculate the gini impurity for each split.

21/4/2025 Lab 6

* Implement Random Forest ensemble method on a given dataset.

- Input:
  - Load the dataset D with n samples and m features.
  - Number of trees to grow T.
  - Number of features to consider at each split F
  - Maximum tree depth MaxDepth.
  - Minimum samples required to split a node minSamplesSplit.

- Step 1: Initialize
  Create an empty list forest = [] to store decision trees.

- Step 2: Build T trees

for t = 1 till T:
  - Bootstrap sampling :-
    Create dataset D_t by sampling n examples with replacement from D.

  - Build a decision tree using D_t
    - At each node of the tree randomly pick F features from all available features

    - For each feature:
      - try different split thresholds
      - calculate the gini impurity for each split.

- Select the best split (with lowest gini)

- Continue recursively for child nodes until:
  - maxDepth is reached
  - MinSamplesSplit is not satisfied.

- Add the trained tree to the forest list

  forest.append(tree)

Step 3: Make Predictions

For each sample x in the test set use majority voting to choose the final label.

Step 4. Evaluate Accuracy

accuracy = $\frac{\text{correct predictions}}{\text{total samples}}$

recall, confusion matrix.

* Implement Boosting ensemble method on dataset

Algorithm (AdaBoost)

Step 1: Initialize weights
Assign equal weight to all training samples

$w_i = \frac{1}{N} \quad \forall \; i \in [1, N]$

Initially, AdaBoost selects a training subset randomly with equal weight

Step 2: Iterate for T rounds (Estimators)
Repeat the process for a pre-defined number of rounds or until no error

Step 3: Train weak Learner
Train a weak classifier on the current dataset using the sample weights

Step 4: Compute weighted Error
Calculate the total error of the classifier with respect to current weights.

Error $\sum w_i \cdot I(y_i \neq h(x_i))$

Step 5. Compute classifier's weight
Assign a weight to the classifier based on its accuracy $\alpha = \frac{1}{2} \log\left(\frac{1-\text{error}}{\text{error}}\right)$

Step 6 Update sample weights
Update the weights of the training samples.
$w_i = w_i \cdot e^{-\alpha y_i h(x_i)}$ then normalize.

Step 7 Voting
Combine all weak learners using their assigned weights.

* Build k-means algorithm to cluster a set of data stored in a .csv file.

Step 1: Load the Dataset.

Step 2: Preprocess the data
Clean the data by handling missing values, normalizing, drop non-numeric columns

Step 3: Choose the number of clusters (K)
You must specify the number of clusters K.

Step 4. Initialize centroids
Randomly select K points from the dataset as initial centroids $c_1 - c_n$

Step 5 Assign data points to closest centroid
- Calculate the distance between each data point and all centroids

Assign each point to the cluster whose centroid is the closest.

Step 6 Update the centroids
For each cluster, compute the mean of the data points assigned to that cl

Step 7 Check for convergence
Check if the centroids have changed fr the previous iteration, if not stop.

Step 8: Repeat the process
Repeat the process of assigning points to the nearest centroid and update still convergence

Step 9: Final cluster
After convergence, each data point will be assigned to one of the K clusters
Label each with its cluster number.

* Implement Dimensionality reduction using PCA method.

Step 1 Load the Dataset

Step 2 Preprocess the data
Handle any missing values in the dataset

Step 3 Compute the Covariance matrix shows how much two random variables vary together.

Step 4 Calculate the Eigenvalues and eigenvectors
understand the variance explained by each component

Step 5. Sort Eigenvalue and Eigenvectors
sort in DO to prioritize the components that explain the most variance

Step 6 Select the top k Eigen vectors
This determines how many dimensions you'll reduce your data to.

Step 7

Project the data onto the new feature

Step 8 Visualize the reduced data

For visualizing purposes, if the data is reduced to 2 dimensions, you can plot the points in a 2D space.

Step 9 Interpret the results

The reduced dataset contains the principal components as new features

21/4/25

Each row in the matrix represents a data point, but now in the lower dimensional space with features