# Project structure explanation:

Testing Commands

Install Dependencies:

pip install scikit-learn numpy

Test the algorithms against the three datasets  
 # Run the script for breast cancer dataset

python breast\_cancer.py

# Run the script for car evaluation dataset

python car\_evaluation.py

# Run the script for Hayes-Roth dataset

python hayes\_roth.py

## Core Scripts

* **breast\_cancer.py**: Script to run classification tasks on the breast cancer dataset using the Custom KNN and Scikit-learn KNN models.
* **car\_evaluation.py**: Script to run classification tasks on the car evaluation dataset.
* **hayes\_roth.py**: Script for classifying the Hayes-Roth dataset using KNN models.
* **mainDriver.py**: Central script for executing tasks across multiple datasets and testing different models.

## Custom Code and Utilities

* **myCustomKNN.py**: The core implementation of a custom K-Nearest Neighbors algorithm from scratch.
* **CBknn.py**: **Class-Based k-Nearest Neighbors (CB-kNN)** algorithm, a variation of the traditional k-Nearest Neighbors (kNN) classifier
* **hyperParamTune.py**: Utilizes grid search to find the optimal values for n\_neighbors, weights, and metrics for the KNN.
* **kFold.py**: Implements K-Fold cross-validation to evaluate the accuracy of the input model dictionary.
* **cleanHelpers.py**: Helper functions for data preprocessing {handling missing values, replacing invalid entries.
* **Scalers.py**: Functions to scale and standardize datasets.
* **hypoTesting.py**: Functions for performing hypothesis testing to compare the statistical significance of the accuracy differences between two models.

KNN:

Finding Neighbors: KNN measures how "close" (similar) new data is to existing data by calculating a distance.

The 'k' in KNN: K is the number of neighbors KNN looks at to make its decision.

Making a Decision: Voting**:** If it's used for classification, KNN looks at the most common category among its neighbors and predicts that.

## CB KNN:

CB-kNN is an extension of KNN designed to handle imbalanced datasets by considering the nearest neighbors from each class separately.

Working:

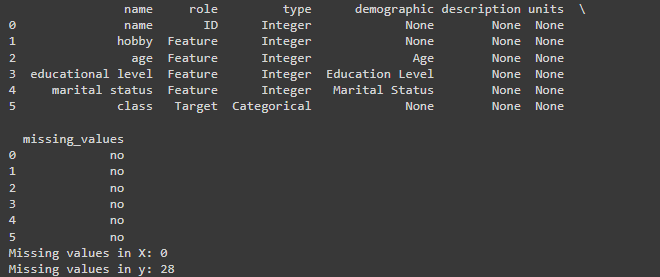
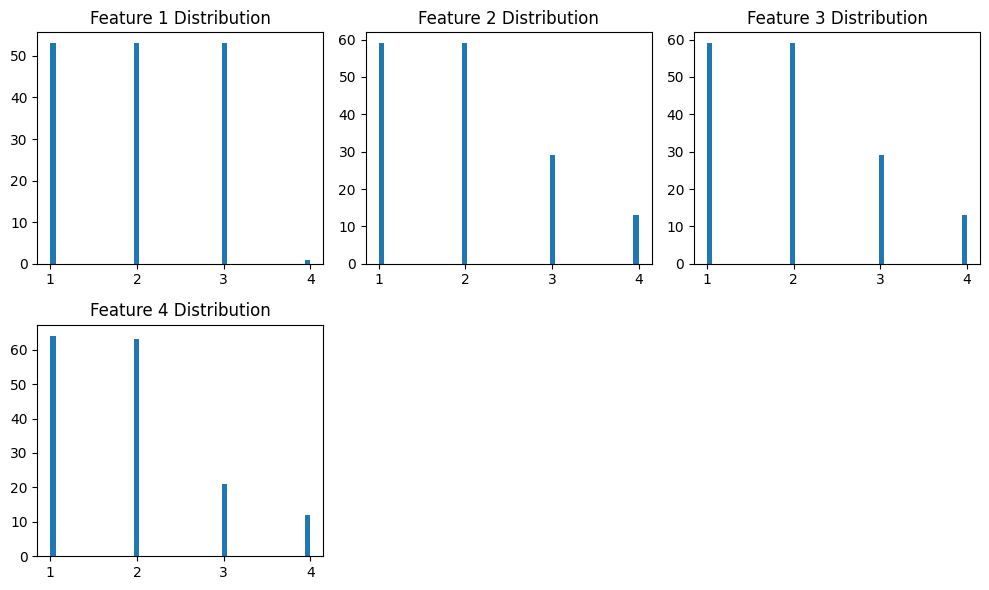
For each class, CB-kNN calculates the k nearest neighbors and computes the harmonic mean of their distances.

The class with the lowest harmonic mean is chosen, emphasizing closer neighbors and reducing the influence of outliers.

# 

# Hayes Roth

## EDA

Did exploratory data analysis to decide the best preprocessing steps  
  
  
  
Data Distribution   


## Preprocessing:

Fixing data:

Replaced null with modes and used standard scaler to scale the data

X, y = process\_dataset(file\_path, column\_names)

X, y = replace\_nan\_and\_question(X, y)

X = replace\_none\_with\_most\_frequent(X)

y = replace\_none\_with\_mode(y)

# Convert features and labels to numeric

X = [[int(value) for value in row] for row in X]

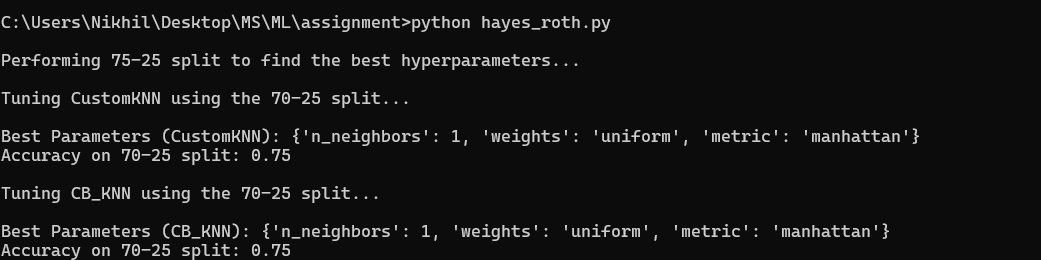
y = [int(float(value)) for value in y]

# Standardize the features

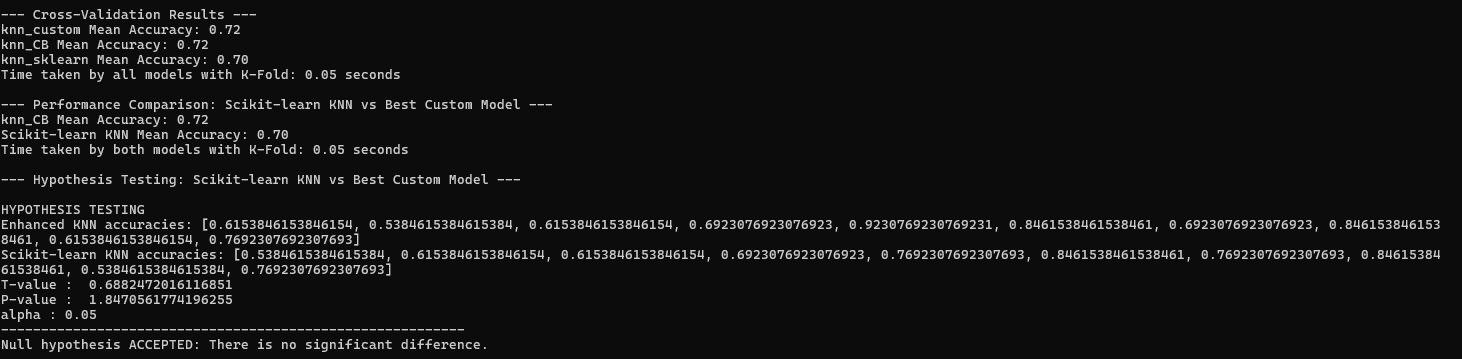
mean, std = standard\_scaler\_fit(X)

X\_scaled = standard\_scaler\_transform(X, mean, std)

## Output

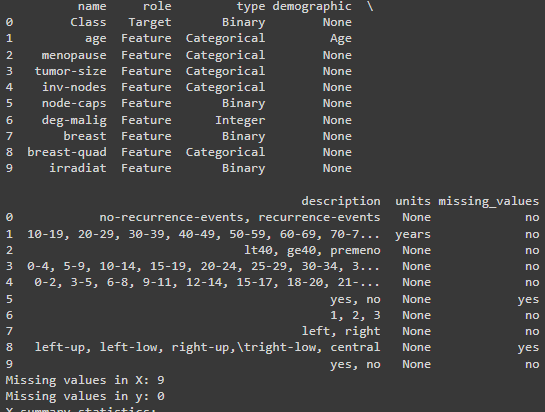


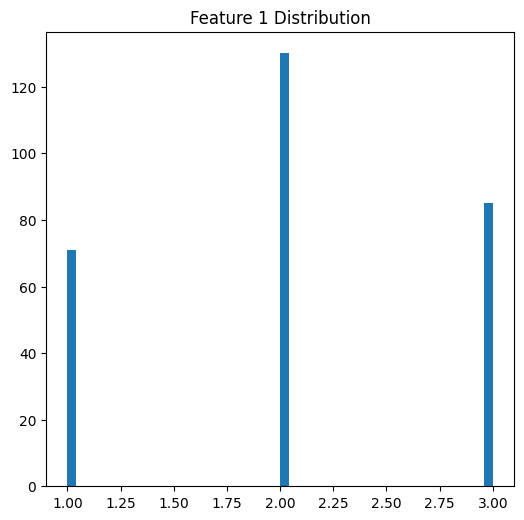
# 

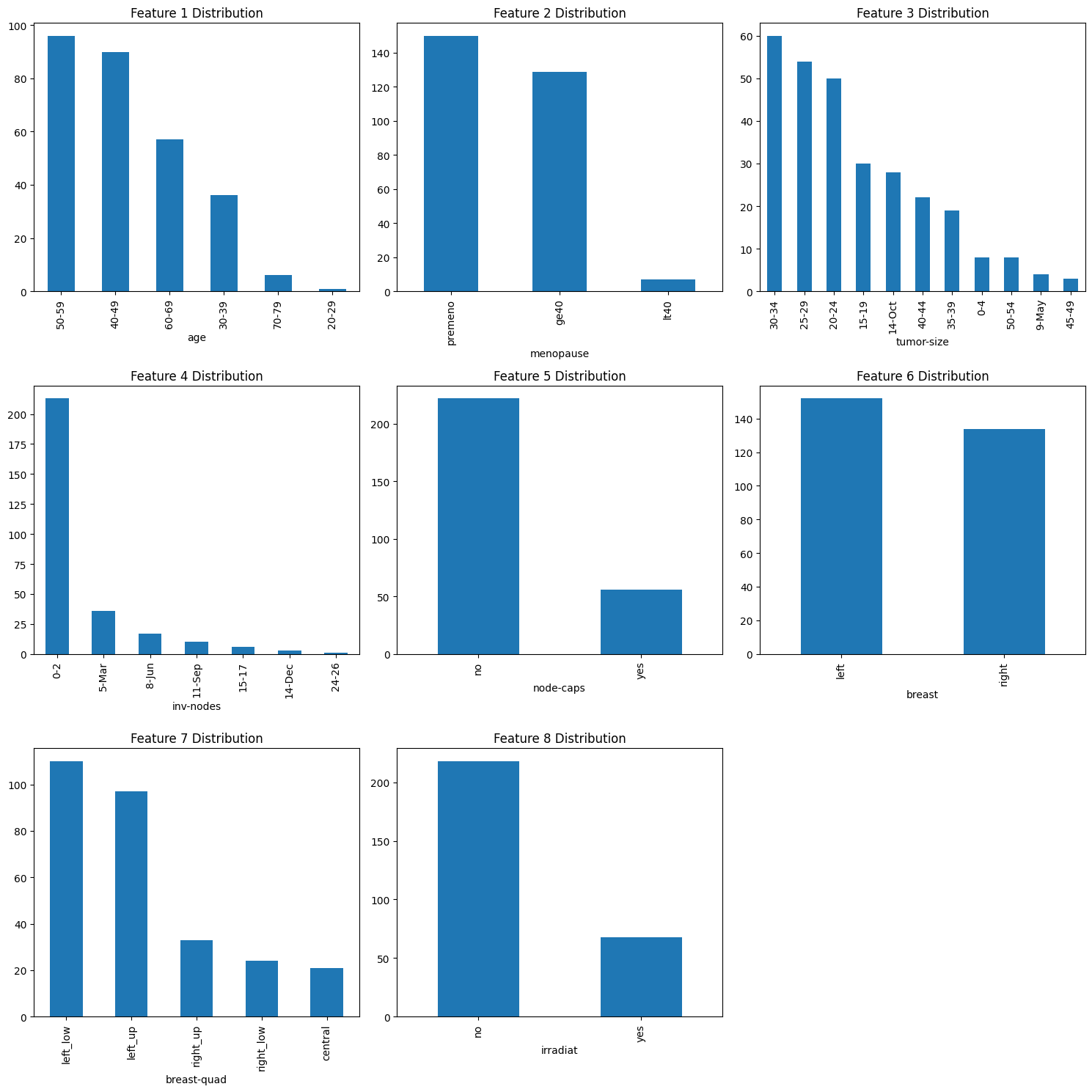


# Breast\_cancer

## EDA







## Preprocessing:

Used the below encodings to convert categorical values to numerical

encodings = {

"recurrence\_status": {"no-recurrence-events": 0, "recurrence-events": 1},

"age": {"10-19": 0, "20-29": 1, "30-39": 2, "40-49": 3, "50-59": 4, "60-69": 5, "70-79": 6},

"menopause": {"lt40": 0, "ge40": 1, "premeno": 2},

"tumor\_size": {"0-4": 0, "5-9": 1, "10-14": 2, "15-19": 3, "20-24": 4, "25-29": 5, "30-34": 6, "35-39": 7, "40-44": 8, "45-49": 9, "50-54": 10},

"inv\_nodes": {"0-2": 0, "3-5": 1, "6-8": 2, "9-11": 3, "12-14": 4, "15-17": 5, "18-20": 6, "21-23": 7, "24-26": 8},

"node\_caps": {"no": 0, "yes": 1, "?": None},

"deg\_malig": {"1": 1, "2": 2, "3": 3},

"breast": {"left": 0, "right": 1},

"breast\_quad": {"left\_up": 0, "left\_low": 1, "right\_up": 2, "right\_low": 3, "central": 4, "?": None},

"irradiation": {"no": 0, "yes": 1}

}

Fixing data:

Replaced null with modes and used standard scaler to scale the data

X, y = process\_dataset(file\_path, column\_names, encodings)

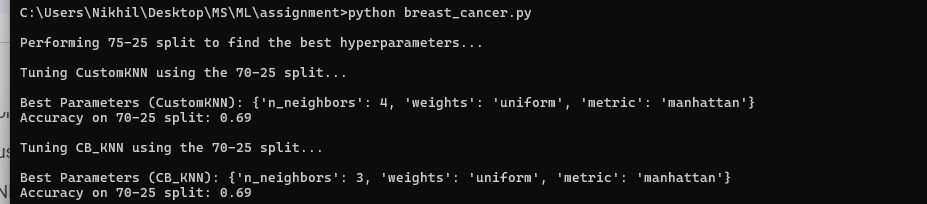
X\_cleaned = replace\_nan\_and\_question(X)

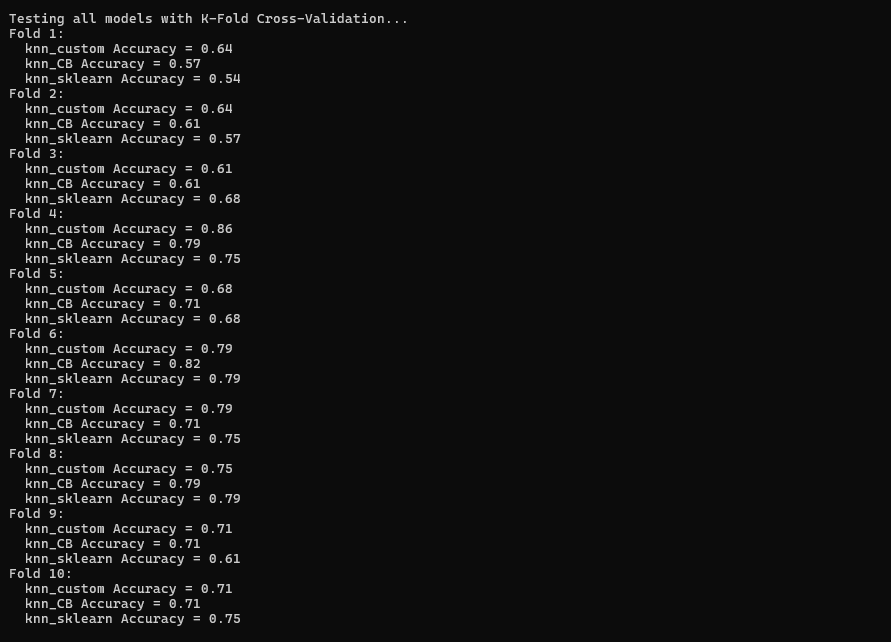
X\_imputed = replace\_none\_with\_most\_frequent(X\_cleaned)

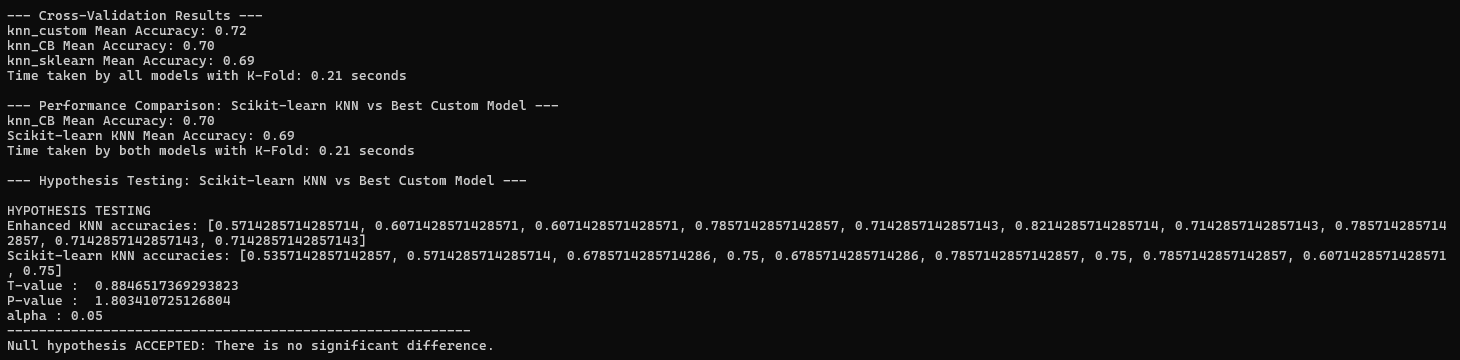
mean, std = standard\_scaler\_fit(X\_imputed)

X\_scaled = standard\_scaler\_transform(X\_imputed, mean, std)

## Output







# 

# Car evaluation:

## EDA

## 

## 

## Preprocessing:

Used the below encodings to convert categorical values to numerical

column\_names = ['buying', 'maint', 'doors', 'persons', 'lug\_boot', 'safety', 'class']

encodings = {

'buying': {'vhigh': 3, 'high': 2, 'med': 1, 'low': 0},

'maint': {'vhigh': 3, 'high': 2, 'med': 1, 'low': 0},

'doors': {'2': 0, '3': 1, '4': 2, '5more': 3},

'persons': {'2': 0, '4': 1, 'more': 2},

'lug\_boot': {'small': 0, 'med': 1, 'big': 2},

'safety': {'low': 0, 'med': 1, 'high': 2},

'class': {'unacc': 0, 'acc': 1, 'good': 2, 'vgood': 3}

}

Fixing data:

Replaced null with modes and used standard scaler to scale the data

X, y = process\_dataset(file\_path, column\_names, encodings)

X\_cleaned = replace\_nan\_and\_question(X)

X\_imputed = replace\_none\_with\_most\_frequent(X\_cleaned)

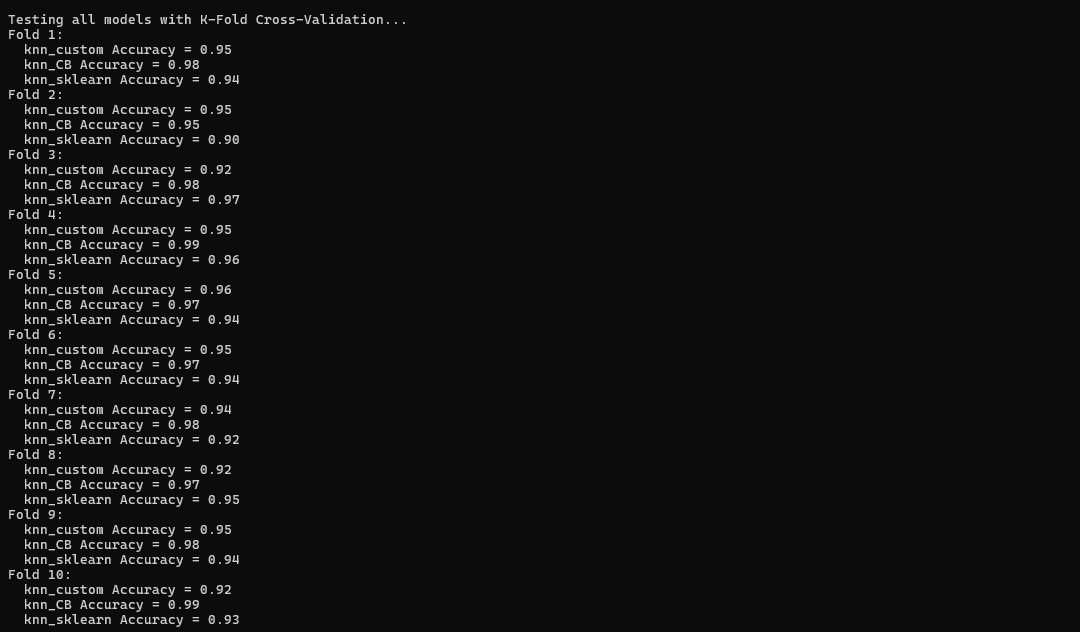
X\_imputed = [[float(value) for value in row] for row in X\_imputed]

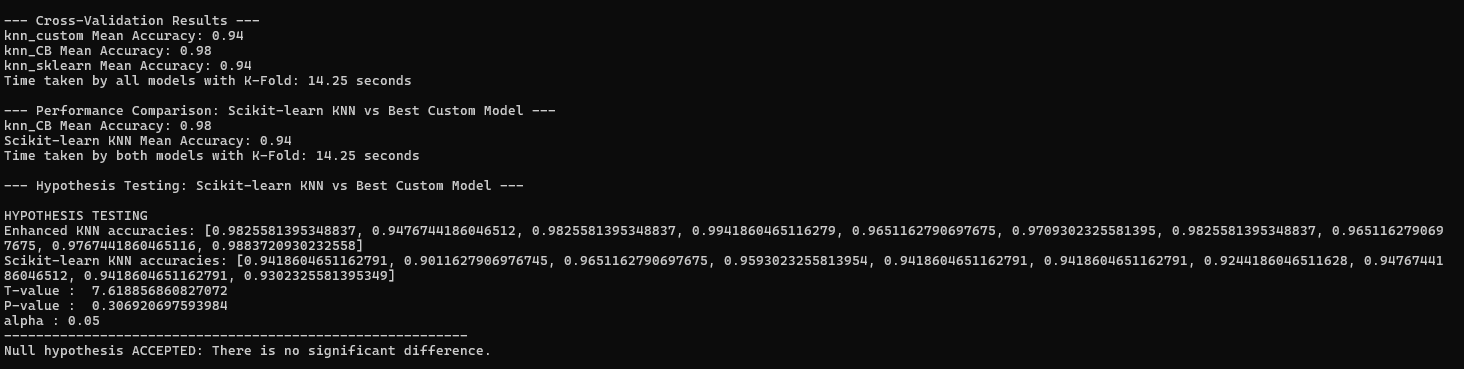
# Standardize the features

mean, std = standard\_scaler\_fit(X\_imputed)

X\_scaled = standard\_scaler\_transform(X\_imputed, mean, std)

## Output





# 

# Findings:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Hayes Roth | Breast Cancer | Car Evaluation |
| Scikit Learn’s KNN | 0.70 | 0.69 | 0.94 |
| Custom KNN | 0.72 | 0.72 | 0.94 |
| Enhanced CBKNN | 0.72 | 0.72 | 0.98 |

The difference between the Sikit learns and the Custom KNN might be due to the Math library used and the parameters were tuned for the custom KNN

Scaling the data is very important as data with a very high range can drastically affect the performance of KNN models.

Hyperparameter selection, such as the distance metric and the number of K neighbors, is significant, as they significantly impact the model's performance.

CBKNN

Unlike KNN, which may favor majority classes, CB-kNN ensures that neighbors from each class are considered independently, giving minority classes a fair chance.

Selects the nearest neighbors from each class separately rather than from the whole dataset, ensuring balanced consideration of all classes.

It uses the harmonic mean to aggregate distances, which emphasizes closer neighbors and reduces the impact of distant ones, improving robustness