**KNN Hyperparameter Tuning with Iris Dataset**

**Author:** Fernando Ian Patricio  
 **Date:** September 2, 2025  
 **Language:** Python  
 **Purpose:** This program finds the optimal number of neighbors (k) for a K-Nearest Neighbors classifier using the Iris dataset. It demonstrates hyperparameter tuning, cross-validation, model evaluation, and visualization of results. The program follows best practices, including structured pseudocode, detailed comments, and reproducible code.

## **Python Pseudocode:**

# Module: KNN Hyperparameter Tuning with Iris Dataset

# Purpose: Use cross-validation to determine the optimal k value for KNN.

#

# Step 1: Import libraries

# numpy -> numerical operations

# matplotlib -> plot accuracy results

# sklearn.datasets -> load Iris dataset

# sklearn.model\_selection -> train\_test\_split, cross\_val\_score

# sklearn.neighbors -> KNeighborsClassifier

#

# Step 2: Load Iris dataset

# - Extract features (X) and labels (y)

#

# Step 3: Split dataset into training (70%) and test (30%) sets

# - Use stratification to maintain class balance

#

# Step 4: Define range of k values (1–30)

#

# Step 5: For each k value:

# - Initialize KNeighborsClassifier with current k

# - Perform 5-fold cross-validation on training data

# - Store mean accuracy score

#

# Step 6: Identify the k with the highest mean accuracy

#

# Step 7: Train final model using best k

# - Fit model on training data

#

# Step 8: Evaluate final model on test set

# - Print accuracy

#

# Step 9: Plot cross-validation accuracy vs. k values

# - Highlight best k with vertical line

## **Python Source Code:**

*"""*

*Module: KNN Hyperparameter Tuning with Iris Dataset*

*Description: This program determines the optimal number of neighbors (k)*

*for a KNN classifier using cross-validation. It evaluates*

*different k values on the Iris dataset and visualizes*

*accuracy results.*

*Author: Fernando Ian Patricio*

*Date: September 2, 2025*

*"""*

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.neighbors import KNeighborsClassifier

# Load Iris dataset (150 samples, 4 features, 3 classes)

iris = load\_iris()

X, y = iris.data, iris.target

# Split data: 70% training, 30% testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=42, stratify=y

)

# Define candidate k values (neighbors)

k\_values = range(1, 31)

accuracy\_scores = []

# Evaluate each k using cross-validation

for k in k\_values:

knn = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(knn, X\_train, y\_train, cv=5) # 5-fold CV

accuracy\_scores.append(scores.mean())

# Find best k

best\_k = k\_values[np.argmax(accuracy\_scores)]

best\_score = max(accuracy\_scores)

print("========================================")

print(f"Optimal number of neighbors (k): {best\_k}")

print(f"Best cross-validation accuracy: {best\_score:.4f}")

# Train final model using best k

final\_knn = KNeighborsClassifier(n\_neighbors=best\_k)

final\_knn.fit(X\_train, y\_train)

# Evaluate on test set

test\_accuracy = final\_knn.score(X\_test, y\_test)

print(f"Test set accuracy with k={best\_k}: {test\_accuracy:.4f}")

print("========================================")

# Plot accuracy results

plt.figure(figsize=(8, 5))

plt.plot(k\_values, accuracy\_scores, marker="o", linestyle="-")

plt.title("KNN Hyperparameter Tuning on Iris Dataset")

plt.xlabel("Number of Neighbors (k)")

plt.ylabel("Cross-Validation Accuracy")

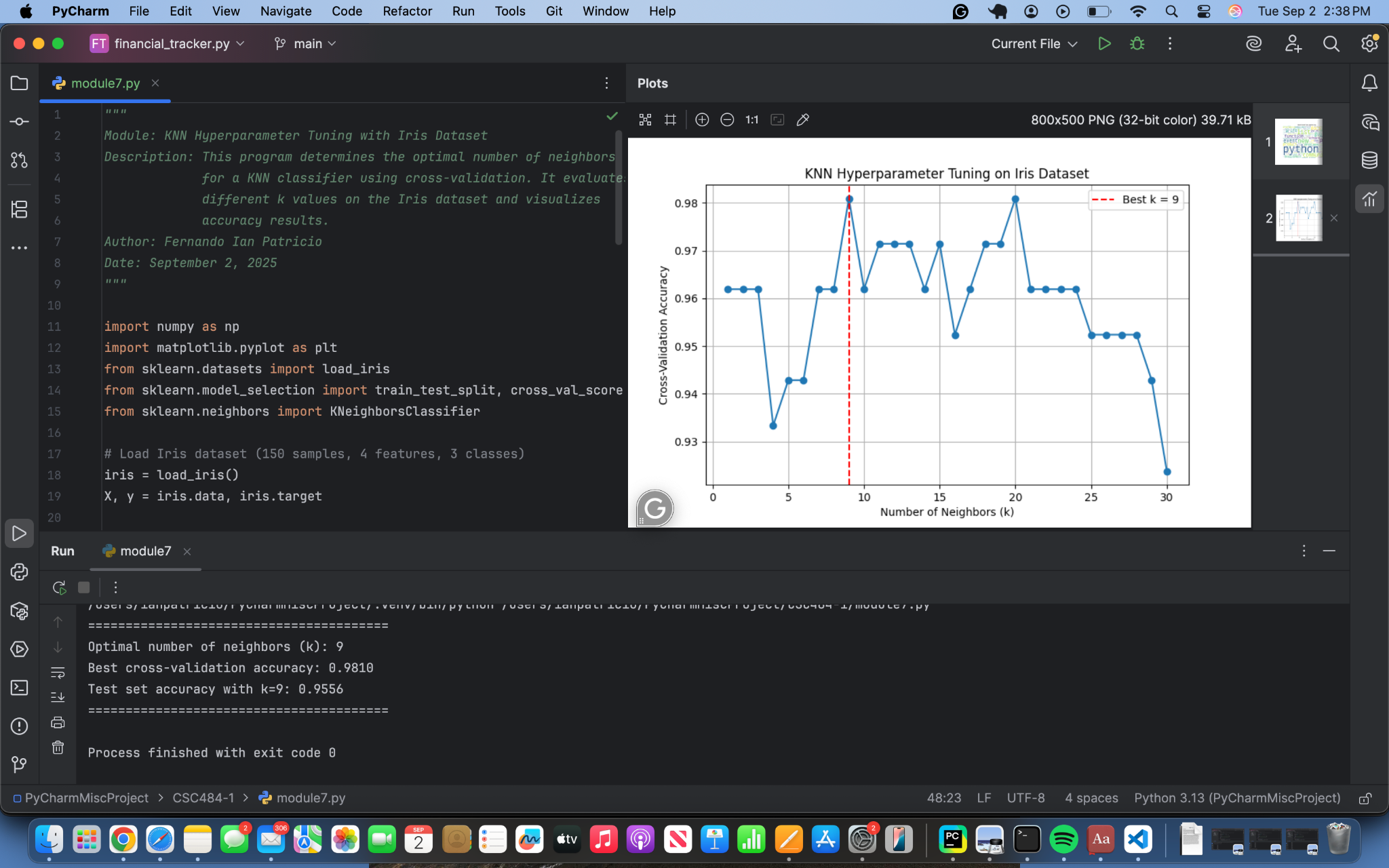
plt.grid(True)

plt.axvline(x=best\_k, color="red", linestyle="--", label=f"Best k = {best\_k}")

plt.legend()

plt.show()

## **Screenshot:**



## **Git Repository:**

## <https://github.com/ianpatricio-csuglobal/CSC484-1>