# **Diabetes Prediction using Neural Networks**

**Author:** Fernando Ian Patricio  
 **Date:** September 7, 2025  
 **Language:** Python  
 **Purpose:** This program builds a neural network to predict whether a patient has diabetes using the Pima Indians Diabetes dataset. It demonstrates the full ML workflow: data preprocessing, model building, training, evaluation, prediction, and visualization. The program follows best practices, including structured pseudocode, detailed comments, and reproducible code.

## **Python Pseudocode**

# Module: Diabetes Prediction with Neural Networks

# Purpose: Predict diabetes using a neural network classifier.

# Step 1: Import libraries

# - pandas -> data handling

# - matplotlib -> visualization

# - sklearn.model\_selection -> train\_test\_split

# - sklearn.preprocessing -> StandardScaler

# - sklearn.metrics -> confusion\_matrix, classification\_report

# - keras -> Sequential, Dense, Input layers

# Step 2: Load Pima Indians Diabetes dataset

# - Assign column names

# - Read local CSV file

# Step 3: Explore data

# - Display first few rows

# - Check info, nulls, shape

# Step 4: Split data into features (X) and target (y)

# - Train/test split (80/20)

# Step 5: Preprocess features

# - Apply StandardScaler normalization

# Step 6: Define neural network

# - Input layer: dimension = number of features

# - Hidden layer 1: 16 neurons, ReLU

# - Hidden layer 2: 8 neurons, ReLU

# - Output layer: 1 neuron, sigmoid

# Step 7: Compile model

# - Optimizer: Adam

# - Loss: binary\_crossentropy

# - Metrics: accuracy

# Step 8: Train model

# - Epochs = 100

# - Batch size = 16

# - Validation split = 20%

# - Save training history

# Step 9: Evaluate model on test set

# - Print loss and accuracy

# Step 10: Make predictions

# - Predict probabilities

# - Convert to binary predictions

# Step 11: Assess performance

# - Confusion matrix

# - Classification report

# Step 12: Visualize training

# - Plot accuracy over epochs

# - Plot loss over epochs

## **Python Source Code**

*"""*

*=====================================================*

*Diabetes Prediction using Neural Networks (Keras)*

*Author: Fernando Ian Patricio*

*Date: September 7, 2025*

*=====================================================*

*Project Description:*

*--------------------*

*This program builds a machine learning model to predict*

*whether a patient has diabetes based on health indicators*

*from the Pima Indians Diabetes dataset. It demonstrates a*

*complete ML workflow:*

*- Data loading & preprocessing*

*- Model definition & training*

*- Evaluation on test data*

*- Visualization of training progress*

*Dataset Source:*

*---------------*

*Pima Indians Diabetes Dataset*

*https://www.openml.org/search?type=data&sort=runs&id=43483&status=active*

*Target Variable:*

*----------------*

*Outcome (0 = No diabetes, 1 = Diabetes)*

*=====================================================*

*"""*

# -----------------------

# 1. Import Libraries

# -----------------------

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion\_matrix, classification\_report

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Input

# -----------------------

# 2. Load and Explore Data

# -----------------------

# Column names for the dataset

column\_names = [

"Pregnancies", "Glucose", "BloodPressure", "SkinThickness",

"Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"

]

# Load dataset (downloaded and saved locally)

df = pd.read\_csv("PimaIndiansDiabetes.csv", header=None, names=column\_names)

print("===== Dataset Preview =====")

print(df.head(), "\n")

print("===== Dataset Info =====")

print(df.info(), "\n")

# -----------------------

# 3. Prepare Features and Target

# -----------------------

X = df.drop("Outcome", axis=1) # Features

y = df["Outcome"] # Target

# Split into train/test sets

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

X, y, test\_size=0.2, random\_state=42

)

# Standardize features for neural network

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# -----------------------

# 4. Build Neural Network

# -----------------------

model = Sequential([

Input(shape=(X\_train.shape[1],)), # Explicit Input layer

Dense(16, activation='relu'), # Hidden layer 1

Dense(8, activation='relu'), # Hidden layer 2

Dense(1, activation='sigmoid') # Output layer (binary classification)

])

# Compile model

model.compile(

optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy']

)

# -----------------------

# 5. Train the Model

# -----------------------

history = model.fit(

X\_train, y\_train,

validation\_split=0.2,

epochs=100,

batch\_size=16,

verbose=1

)

# -----------------------

# 6. Evaluate the Model

# -----------------------

loss, accuracy = model.evaluate(X\_test, y\_test, verbose=0)

print("===== Model Evaluation on Test Data =====")

print(f"Test Loss : {loss:.4f}")

print(f"Test Accuracy : {accuracy:.4f}\n")

# -----------------------

# 7. Make Predictions

# -----------------------

y\_pred\_prob = model.predict(X\_test)

y\_pred = (y\_pred\_prob > 0.5).astype(int)

print("===== Confusion Matrix =====")

print(confusion\_matrix(y\_test, y\_pred), "\n")

print("===== Classification Report =====")

print(classification\_report(y\_test, y\_pred))

# -----------------------

# 8. Visualize Training History

# -----------------------

plt.figure(figsize=(12, 5))

# Accuracy curve

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.title('Training vs Validation Accuracy')

plt.legend()

# Loss curve

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epoch')

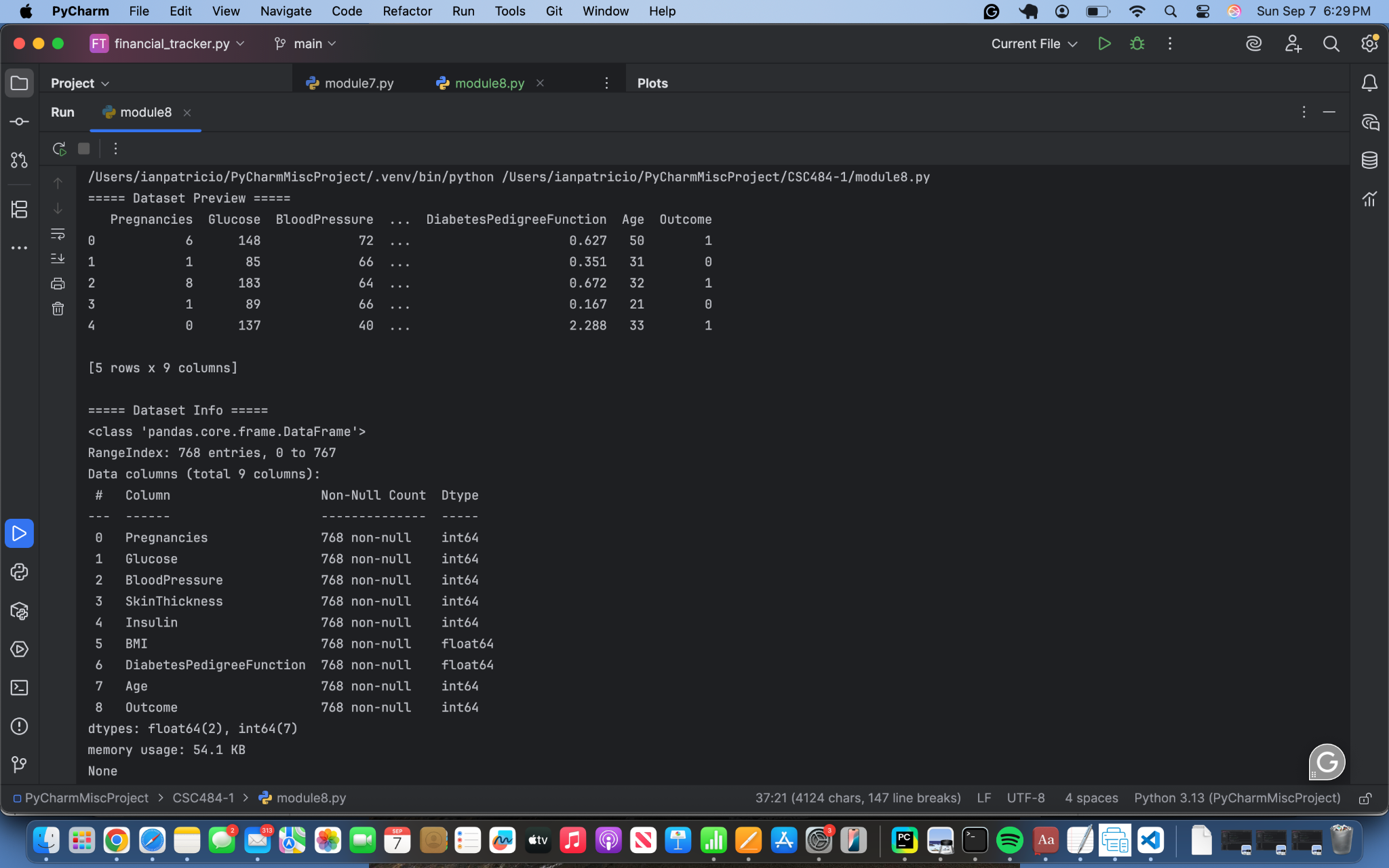
plt.ylabel('Loss')

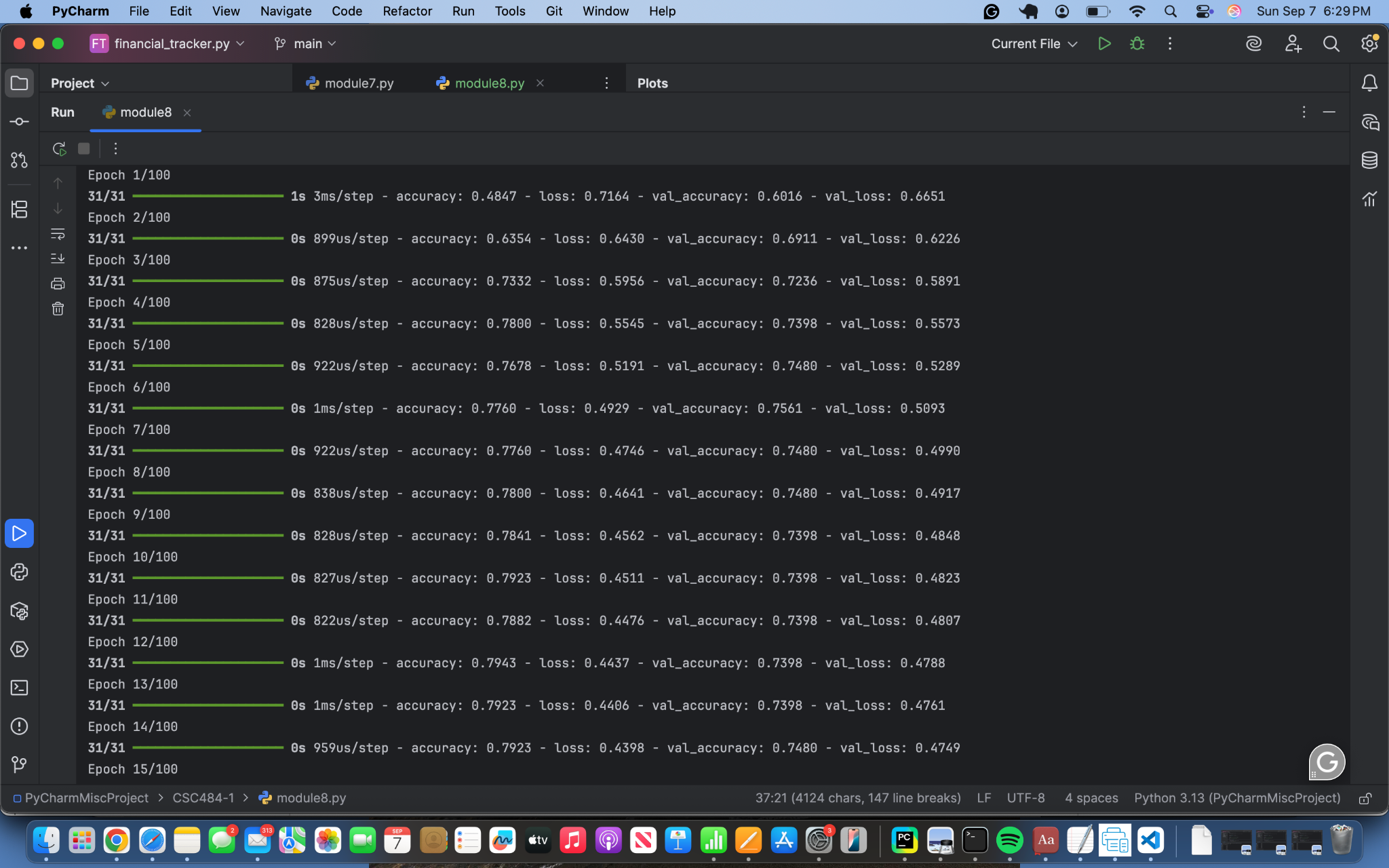
plt.title('Training vs Validation Loss')

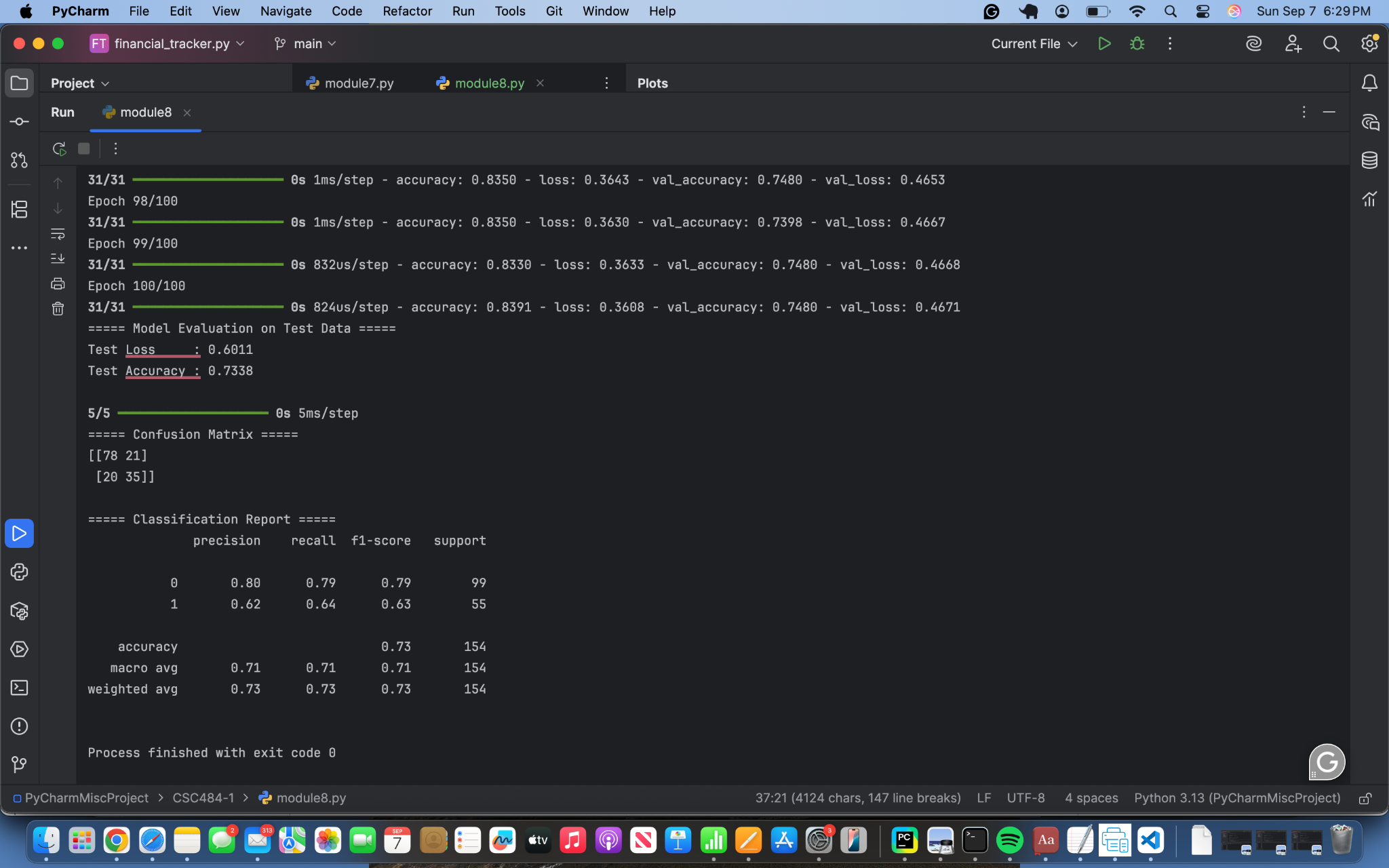
plt.legend()

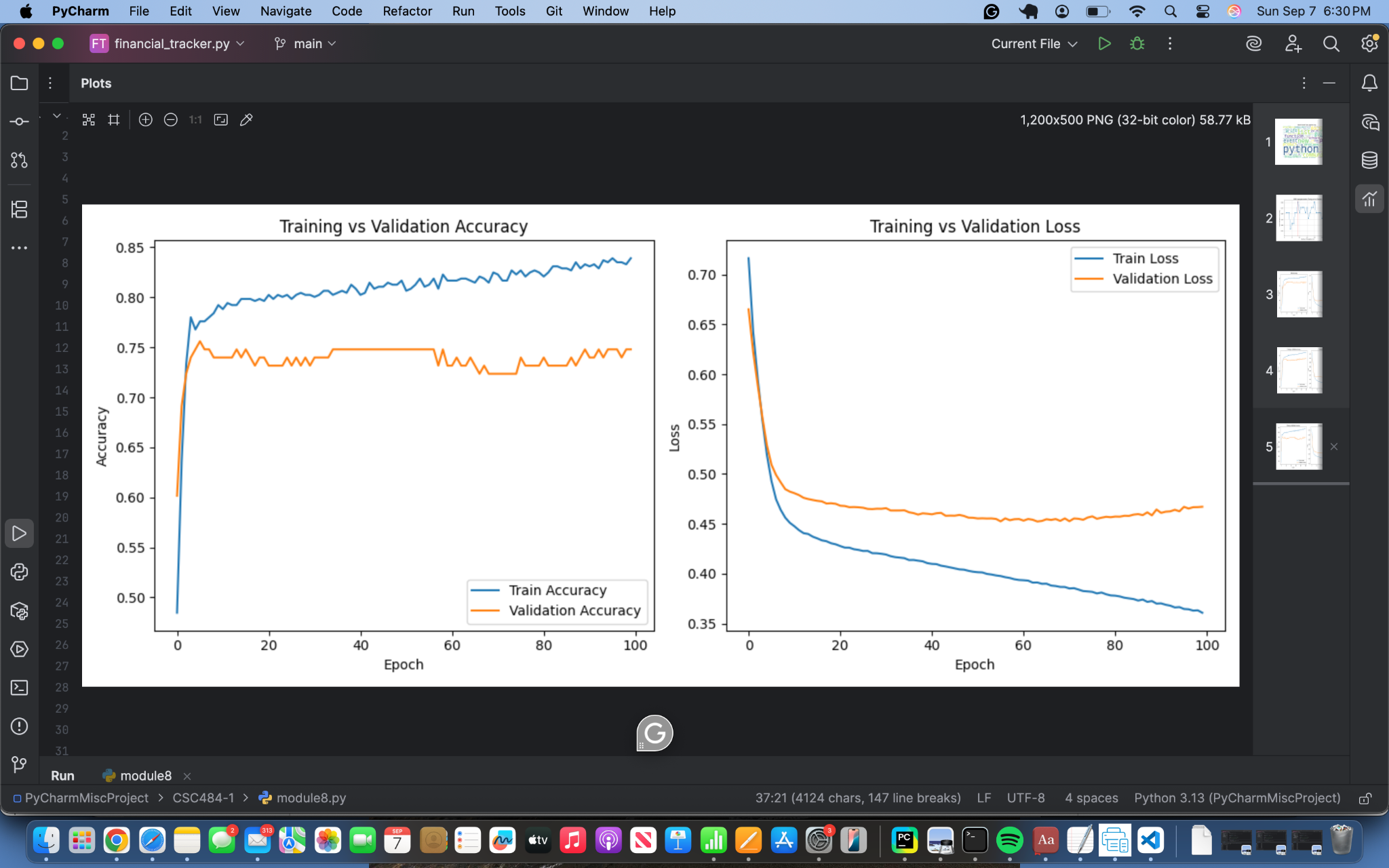
plt.tight\_layout()

plt.show()

**Screenshots**







**Git Repository**<https://github.com/ianpatricio-csuglobal/CSC484-1>