Title: Documentation of Pickle File Creation and Testing for Heart Disease Prediction Model

---

Table of Contents

---

1. Introduction

This document provides a detailed explanation of the code used to create a pickle file containing a logistic regression model for predicting heart disease. It also covers how to test this model using new input data. The use of pickle in Python allows for the serialization and deserialization of Python objects, making it easier to save and load machine learning models.

2. Project Overview

The project aims to build a logistic regression model using a dataset on heart disease. After training the model, it is saved into a pickle file, which can be later used to make predictions on new data. This process is crucial for deploying machine learning models in real-world applications, allowing users to load the model without retraining it.

3. Code Explanation

3.1. Creating the Pickle File

The first part of the code involves importing necessary libraries, loading the dataset, preprocessing the data, training the model, and saving it as a pickle file.

```
CODE :-
--------------------------------------------------------------------------------
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import pickle
```

```
# Load and inspect the dataset
heart_data = pd.read_csv('heart.csv')
print(heart_data.head())

# Preprocessing: Encoding categorical variables and scaling continuous variables
heart_data_x = heart_data.drop("HeartDisease", axis=1)
heart_data_y = heart_data['HeartDisease']

# One-hot encode categorical variables
heart_data_x_encoded = pd.get_dummies(heart_data_x, drop_first=True).astype(int)

# Standard scaling for continuous features
continuous_features = ["Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak"]
scaler = StandardScaler()
heart_data_x_encoded[continuous_features] =
scaler.fit_transform(heart_data_x_encoded[continuous_features])

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(heart_data_x_encoded,
heart_data_y, test_size=0.3, random_state=42)

# Define and train Logistic Regression model
logreg_model = LogisticRegression(max_iter=1000)
logreg_model.fit(X_train, y_train)

# Save the model and scaler into a pickle file
with open('heart_disease_logreg_model.pkl', 'wb') as file:
    pickle.dump({
        'logistic_regression': logreg_model,
        'scaler': scaler,
        'features': heart_data_x_encoded.columns.tolist()
    }, file)

--------------------------------------------------------------------------------
```

Explanation:

- **Library Imports**: Essential libraries for data manipulation (`pandas`, `numpy`), model building (`sklearn`), and saving the model (`pickle`) are imported.
- **Dataset Loading**: The heart disease dataset is loaded using `pd.read_csv`.
- **Data Preprocessing**:
  - The target variable (`HeartDisease`) is separated from the features.
  - Categorical variables are encoded using one-hot encoding to create binary columns.
  - Continuous features are scaled using `StandardScaler` to normalize their values.
- **Data Splitting**: The dataset is divided into training and testing sets using an 70/30 split.
- **Model Training**: A logistic regression model is defined and trained on the training dataset.
- **Saving the Model**: The model, scaler, and feature names are saved into a pickle file for future use.

3.2. Testing the Pickle File

The second part of the code demonstrates how to load the saved model and make predictions on new input data.

CODE :-

```
--------------------------------------------------------------------------------
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import pickle

# Define continuous features
continuous_features = ["Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak"]

# Create a new sample dataset
new_data = {
    'Age': [45, 60, 72],
    'Sex': ['M', 'F', 'M'],
    'ChestPainType': ['ATA', 'NAP', 'ASY'],
    'RestingBP': [120, 140, 130],
    'Cholesterol': [240, 200, 280],
    'FastingBS': [0, 1, 0],
    'RestingECG': ['Normal', 'ST', 'LVH'],
    'MaxHR': [150, 130, 100],
    'ExerciseAngina': ['N', 'Y', 'Y'],
    'Oldpeak': [1.0, 2.3, 3.5],
    'ST_Slope': ['Up', 'Flat', 'Down']
}

# Convert the new data to a DataFrame
new_data_df = pd.DataFrame(new_data)

# Load the saved model and scaler from the pickle file
with open('heart_disease_logreg_model.pkl', 'rb') as file:
    saved_objects = pickle.load(file)

# Access the scaler and model
scaler = saved_objects['scaler']
logreg_model = saved_objects['logistic_regression']
model_features = saved_objects['features']

# Preprocess the new data (one-hot encoding and scaling)
new_data_encoded = pd.get_dummies(new_data_df, drop_first=True).astype(int)
new_data_encoded[continuous_features] =
scaler.transform(new_data_encoded[continuous_features])

# Ensure that the new dataset has the same columns as the training data
for col in model_features:
    if col not in new_data_encoded.columns:
        new_data_encoded[col] = 0  # Add missing columns with default 0

new_data_encoded = new_data_encoded[model_features]  # Reorder columns to match
the training set

# Make predictions using the loaded Logistic Regression model
logreg_predictions = logreg_model.predict(new_data_encoded)

# Output the predictions
print("Logistic Regression Predictions:", logreg_predictions)

--------------------------------------------------------------------------------
```

Explanation:

- **New Sample Data Creation**: A new dataset mimicking the structure of the original dataset is created for testing.
- **Loading the Model**: The previously saved model and scaler are loaded from the pickle file.
- **Preprocessing New Data**: The new dataset undergoes one-hot encoding and scaling of continuous features to match the preprocessing done during training.
- **Column Alignment**: The columns of the new dataset are ensured to match the training dataset, adding any missing columns with a default value of 0.
- **Making Predictions**: The logistic regression model is used to predict heart disease on the new data, and the results are printed.

4. Conclusion

This project demonstrates the entire process of building, saving, and testing a logistic regression model for heart disease prediction using Python's `pickle` module. The ability to save and load models enables efficient use of machine learning in real-world applications without the need to retrain models.

5. References

- Python documentation for pickle
- Scikit-learn Documentation
- Pandas Documentation