Diabetes Prediction System Documentation

Introduction

This documentation outlines the development of a diabetes prediction system. The goal is to select a suitable machine learning algorithm, train the model, and evaluate its performance using the provided dataset named 'diabetes.csv'.

Phase 4: Development Part 2

In this phase, we continue building the diabetes prediction system by selecting a machine learning algorithm, training the model, and evaluating its performance.

Code Components

The code consists of several key components:

1. Importing Necessary Libraries

- Key Python libraries are imported for data manipulation, model building, and performance evaluation. These libraries include NumPy, pandas, scikit-learn, and related modules.

2. Loading the Dataset

- The code loads the pre-processed dataset from a CSV file named 'diabetes.csv'.

3. Splitting the Data

- The dataset is split into feature data (X) and the target variable 'Pregnancies'.

4. Model Initialization and Training

- A Logistic Regression model is initialized and trained using the feature data (X_train) and the target variable 'Pregnancies'.

5. Making Predictions

- The trained model is used to make predictions on the test set (X_test).

6. Calculating Evaluation Metrics

- Several evaluation metrics are calculated to assess the model's performance. The metrics include accuracy, precision, recall, and F1 score.

7. Handling Multiclass Classification

- The code example provided can handle multiclass classification problems.

8. Error Handling

- The code includes error handling for common issues, such as precision/recall in multiclass classification and convergence warnings for the Logistic Regression algorithm.

9. Output

- The calculated evaluation metrics, including accuracy, precision, recall, and F1 score, are printed to the console.

```python
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix

data = pd.read_csv('diabetes.csv')

X = data.drop('Pregnancies', axis=1)
y = data['Pregnancies']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy}')
print(f'Precision (weighted): {precision}')
print(f'Recall (weighted): {recall}')
print(f'F1 Score (weighted): {f1}')
```

```
Accuracy: 0.15584415584415584
Precision (weighted): 0.10758894233233751
Recall (weighted): 0.15584415584415584
F1 Score (weighted): 0.11096118034397631
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Usage

To use this code as a basis for building a diabetes prediction system:

1. Ensure that you have a pre-processed dataset named 'diabetes.csv'.
2. The code assumes that 'Pregnancies' is the target variable in your dataset. Adjust the target variable name if needed.
3. Customize the machine learning algorithm if necessary.
4. Run the code to train the model and evaluate its performance.

Conclusion

This documentation and accompanying code provide a foundation for developing a diabetes prediction system. Users can adapt and expand upon this code to suit their specific dataset and requirements. It's crucial to choose the most appropriate machine learning algorithm and evaluation metrics for the task at hand.