Department of Information Technology – University of the Punjab

Programming for AI – MPhil/PhD (AI) F22

Lab-08

Max Time: 2.5 hours   Date: 15-02-2023

Instructions:

- Please provide your own solutions and DO NOT COPY the code from your colleagues or the web.
- You can discuss your problems only with the teachers.
- All tasks carry equal points.

Task # 01 - Linear Regression [8 x 5 = 40]

Problem Statement

Write a Python function get_diabetes_predictions(feature_idx) that takes an integer feature_idx as input, selects the input feature at index feature_idx from the diabetes dataset, trains a linear regression model using the selected feature, makes predictions on the testing set, and returns the model, mean squared error and coefficient of determination of the model on the testing set.

Signature

```
def get_diabetes_predictions(feature_idx:int) -> Tuple[LinearRegression, float, float]:
  pass
```

Input

- feature_idx (1 <= feature_idx <= 10): An integer representing the index of the input feature to use for training the model.

Output

- A tuple containing the following elements:
  - LinearRegression: The trained linear regression model.
  - float: The mean squared error of the model on the testing set.
  - float: The coefficient of determination (R-squared score) of the model on the testing set.

Department of Information Technology – University of the Punjab

Programming for AI – MPhil/PhD (AI) F22

Lab-08

Instructions

Your program should perform the following steps:

1. Load the diabetes dataset using Scikit-learn's datasets.load_diabetes function.
2. Split the dataset into training and testing sets using scikit-learn's train_test_split function. Use 80% of the data for training and 20% for testing.
3. Select the input feature at index feature_idx from the dataset.
4. Create a linear regression model using scikit-learn's LinearRegression class.
5. Fit the model to the training data using the fit method.
6. Make predictions on the testing data using the predict method.
7. Calculate the mean squared error (MSE) and R-squared score of the model on the testing data using scikit-learn's mean_squared_error and r2_score functions.
8. Return the trained model, MSE and R-squared score as a tuple.

Example Usage

```
model, mse, r2 = get_diabetes_predictions(2)
print("Mean squared error: {:.2f}".format(mse))
print("R-squared score: {:.2f}".format(r2))
```

Department of Information Technology – University of the Punjab

Programming for AI – MPhil/PhD (AI) F22

Lab-08

Task # 02 - MLP Classification [7x5 = 35]

Problem Statement

Write a Python function get_diabetes_mlp_predictions(hidden_layers) that takes a tuple hidden_layers as input, representing the number of hidden layers and number of neurons in each hidden layer of an MLPClassifier. The function should train the MLPClassifier on the diabetes dataset, make predictions on the testing set, and return the model, mean squared error and coefficient of determination of the model on the testing set.

Signature

```
def get_diabetes_mlp_predictions(hidden_layers: Tuple[int]) -> Tuple[MLPClassifier, float, float]:
  pass
```

Input

- hidden_layers (1 <= len(hidden_layers) <= 5): A tuple representing the number of hidden layers and number of neurons in each hidden layer of the MLPClassifier. The tuple should contain between 1 and 5 integers, where each integer represents the number of neurons in that hidden layer. For example, (10, 5) would represent an MLPClassifier with 2 hidden layers, the first with 10 neurons and the second with 5 neurons.

Output

- A tuple containing the following elements:
  - MLPClassifier: The trained MLPClassifier model.
  - float: The mean squared error of the model on the testing set.
  - float: The coefficient of determination (R-squared score) of the model on the testing set.

Instructions

Your program should perform the following steps:

1. Load the diabetes dataset using Scikit-learn's datasets.load_diabetes function.
2. Split the dataset into training and testing sets using scikit-learn's train_test_split function. Use 80% of the data for training and 20% for testing.
3. Create an MLPClassifier using scikit-learn's MLPClassifier class. Use the

hidden_layers tuple to specify the number of hidden layers and number of neurons in each hidden layer.

4. Fit the model to the training data using the fit method.
5. Make predictions on the testing data using the predict method.
6. Calculate the mean squared error (MSE) and R-squared score of the model on the testing data using scikit-learn's mean_squared_error and r2_score functions.
7. Return the trained model, MSE and R-squared score as a tuple.

Example Usage

```
model, mse, r2 = get_diabetes_mlp_predictions((10, 5))
print("Mean squared error: {:.2f}".format(mse))
print("R-squared score: {:.2f}".format(r2))
```