AI in Bio-robotics

Task 1:
Linear regression

Prepared By
Esraa Ali

Introduction

The code implements linear regression to predict insurance charges based on the age and BMI (Body Mass Index) of individuals. The dataset used in this code is "insurance.csv"

Code Overview

1- Implementation from scratch

1. Importing the required libraries:
- *pandas* for data manipulation and analysis.
- *matplotlib.pyplot* for data visualization.
- *numpy* for numerical computations.

2. Loading the dataset:
- The insurance dataset is loaded using the *read_csv*

3. Data Normalization:
- The *normalize* function is defined to normalize the feature matrix `X` and the target variable `y`.
- The mean and standard deviation of each feature are computed using `*np.mean*` and `*np.std*` functions.
- The feature matrix `X` and target variable `y` are then normalized by subtracting the mean and dividing by the standard deviation.

4. Linear Regression Functions:
- *fit* function implements the gradient descent algorithm to find the optimal weights and bias for linear regression.
- It takes the normalized feature matrix `X`, normalized target variable `y`, number of iterations, and learning rate as input.
- Initialized weights and bias to zeros
- In each iteration, it computes the predicted values, gradients, and updates the weights and bias using the gradient descent update rule.
- Finally, it returns the optimized weights and bias.
- *predict* function predicts the target variable `y` using the feature matrix `X`, weights, and bias.
- *mean_square_error* computes the mean squared error between the predicted and actual values of `y`.

5. Data Preparation:
- The feature matrix `X` is created by selecting the 'age' and 'bmi' columns from the insurance dataset.
- The target variable `y` is created by selecting the 'charges' column.

6. Data Normalization:
- The feature matrix `X` and target variable `y` are normalized using the *normalize* function.
- Normalization is performed to scale the features and target variable to a similar range, which helps in improving the convergence and stability of the gradient descent algorithm.

7. Model Training:
- The learning rate and number of iterations are set
- The `fit` function is called with the normalized feature matrix `X_norm`, normalized target variable `y_norm`, number of iterations, and learning rate as inputs.
- The function returns the optimized weights and bias.

8. Prediction and Evaluation:
- The *predict* function is called with the normalized feature matrix `X_norm`, weights, and bias to obtain the predicted values of the target variable.
- The *mean_square_error* function is used to compute the mean squared error between the predicted and actual values of `y_norm`.

9. Denormalization:
- The predicted and actual charges are denormalized by multiplying with the standard deviation and adding the mean.

10. Result Display:
- The weights, bias, and mean squared error are printed.
- A scatter plot is created to visualize the predicted charges against the actual charges.

2- Implementation using Scikit-learn library

1. Importing the required libraries:
- *pandas* for data manipulation and analysis.
- *matplotlib.pyplot* for data visualization.
- *sklearn.linear_model.LinearRegression* is used for linear regression modeling.
- *sklearn.preprocessing.StandardScaler* is used for data normalization.

2. Load the dataset:
The insurance dataset is loaded from the 'insurance.csv' file.

3. Prepare the data:
The feature matrix `X` is created by selecting the 'age' and 'bmi' columns, and the target variable `y` is set as the 'charges' column.

4. **Normalize the features:**
   The feature matrix `X` is normalized using the `*StandardScaler*` from scikit-learn.

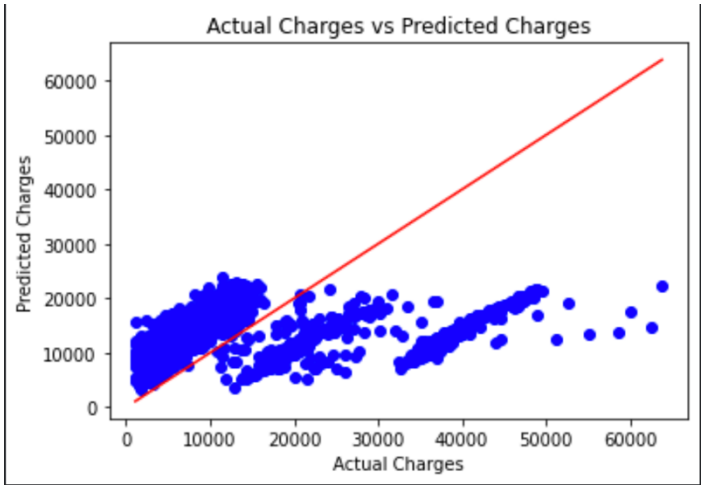
5. **Create the model:**
   A linear regression model is created using the `*LinearRegression*` class from scikit-learn.

6. **Train the model:**
   The model is trained using the normalized feature matrix `X` and target variable `y`.

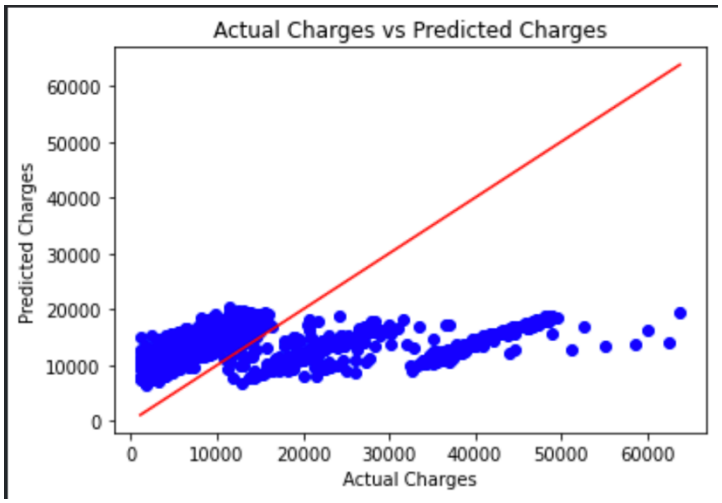
7. **Predict charges:**
   The trained model is used to predict charges using the normalized feature matrix `X`.

8. **Visualize the results:**
   A scatter plot is created to compare the actual charges with the predicted charges.

Comparison between results



using Scikit-learn library



using scratch algorithm

The two figures show that, in addition to being more user-friendly, the scikit-learn library is also more accurate.