**"Introduction to Artificial Intelligence" Experiment report**

|  |
| --- |
| The experimental report gives the content elements in the following order:   1. Flow chart of linear regression algorithm 2. Experimental results and analysis diagram    1. python programming environment is required for experiments, including sklearn, numpy, matplotlib.    2. Implement simple linear regression algorithm predictions using numpy. Ask to explain the code sentence by sentence.    3. sklearn library is used to implement multivariate (using Boston house pricing dataset) linear regression algorithm prediction.    4. Compare the difference between self-written linear regression algorithm and sklearn toolkit, and compare the difference of different parameters in sklearn library    5. Refer to this: [Linear Regression (Python Implementation) - GeeksforGeeks](https://www.geeksforgeeks.org/linear-regression-python-implementation/) (another reference: [Linear Regression Explained, Step by Step (machinelearningcompass.com)](https://machinelearningcompass.com/machine_learning_models/linear_regression/)) 3. Source code and necessary comments 4. Experimental summary and experience   Note: The experiment report is required to be completed independently, and students are allowed to discuss with each other, but plagiarism is absolutely not allowed. Once found, this experiment is recorded as 0 points.  **Linear Regression Algorithm**  **Flow Chart:**  C:\Users\Alex Joshua Chirwa\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\Screenshot (142).jpeg  C:\Users\Alex Joshua Chirwa\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\Screenshot (143).jpeg  C:\Users\Alex Joshua Chirwa\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\Screenshot (144).jpeg  **Simple Linear Regression:**  Single Independent Variable:   * In simple linear regression, there is only one independent variable (feature) that is used to predict the dependent variable. * The relationship between the independent variable and the dependent variable is modeled as a straight line.   Equation:   * The equation for simple linear regression is of the form:   Y = β0 + β1X + C where Y is the dependent variable, X is the independent variable, β0 is the intercept, β1 is the slope, and C is the error term.    Graphical Representation:   * In a simple linear regression, the relationship between the variables can be visualized as a straight line on a 2D scatter plot.   **Multivariate Linear Regression:**  Multiple Independent Variables:   * In multivariate linear regression, there are multiple independent variables used to predict the dependent variable. * The model considers the simultaneous influence of multiple features on the target variable.   Equation:   * The equation for multivariate linear regression is of the form:   Y = β0 + β1X + β2X2 +…+ βnXn + C where Y is the dependent variable, X1,X2,…,Xn are the independent variables, β0 is the intercept, β1, β2,…, βn are the coefficients, and C is the error term.    Graphical Representation:   * Multivariate linear regression involves a multidimensional space, making it challenging to represent the entire relationship graphically. However, partial dependence plots or 3D plots can be used to visualize the relationship between individual independent variables and the dependent variable.   **Experimental Results and Analysis**  **Python Programming Environment Setup:**  Imported Libraries:  # Import necessary libraries import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression # from sklearn.metrics import mean\_squared\_error from sklearn import metrics import seaborn as sns import matplotlib.pyplot as plt  **Implementation of Simple Linear Regression Algorithm with Numpy:**  x = df[['area', 'room number']] y = df['house price']  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.4, random\_state=42)  model = LinearRegression()  model.fit(x\_train, y\_train)  y\_pred = model.predict(x\_test)   * I used pandas to read the dataset and selected features (‘area’, and ‘room number’) and the target variable (‘house price’). * Data is split into training and testing sets using ‘train\_test\_split’. * A linear regression model is created and trained using the training data. * Predictions are made on the test data.   **Sklearn Library for Multivariate Linear Regression:**  This part of the code implements multivariate linear regression using the Sklearn library.  **Comparison and Parameter Analysis:**  plt.scatter(x\_test['area'], y\_test, color='red', label='Actual Prices') plt.scatter(x\_test['area'], y\_pred, color='black', label='Predicted Prices') plt.xlabel('Area') plt.ylabel('House Price') plt.title('Linear Regression Demo') plt.legend() plt.show()   * The code visualizes the actual vs predicted house prices for the ‘area’ feature. * This provides a comparison between the actual and predicted values.   print('MAE', metrics.mean\_absolute\_error(y\_test, y\_pred)) print('MSE', metrics.mean\_squared\_error(y\_test, y\_pred)) print('RMSE', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))   * I calculated and printed Mean Absolute Error(MAE), Mean Squared Error(MSE), and Root Mean Squared Error(RMSE). * This evaluates the performance of the linear regression model.   **Full Source Code:**  # Import necessary libraries import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression # from sklearn.metrics import mean\_squared\_error from sklearn import metrics import seaborn as sns import matplotlib.pyplot as plt   file\_path = "C:\\Users\\Alex Joshua Chirwa\Desktop\Artificial Intelligence\house data.csv" # Read the dataset from the CSV file df = pd.read\_csv(file\_path) print(df.head())  x = df[['area', 'room number']] y = df['house price']  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.4, random\_state=42)  model = LinearRegression()  model.fit(x\_train, y\_train)  y\_pred = model.predict(x\_test)  #plt.scatter(y\_test,y\_pred)  score = model.score(x\_test, y\_test) print('Model R^2 Score:', score)  plt.scatter(x\_test['area'], y\_test, color='red', label='Actual Prices') plt.scatter(x\_test['area'], y\_pred, color='black', label='Predicted Prices') plt.xlabel('Area') plt.ylabel('House Price') plt.title('Linear Regression Demo') plt.legend() plt.show()  print('MAE', metrics.mean\_absolute\_error(y\_test, y\_pred)) print('MSE', metrics.mean\_squared\_error(y\_test, y\_pred)) print('RMSE', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))  **Result:**  C:\Users\Alex Joshua Chirwa\Pictures\Screenshots\Screenshot (149).png  area room number house price  0 2104 3 399900  1 1600 3 329900  2 2400 3 369000  3 1416 2 232000  4 3000 4 539900  Model R^2 Score: 0.6620314861441741  MAE 72186.35151737282  MSE 8134843351.963599  RMSE 90193.36645210444  Process finished with exit code 0  **Explanation:**   1. Model R^2 Score:  * R^2(coefficient of determination) is a measure of how well the model explains the variance in the target variable. * The model achieved an R^2 score of 0.662, indicating that around 66.2% of the variability in house prices is explained by the model.  1. MAE(Mean Absolute Error):  * MAE represents the average absolute difference between the actual and predicted values * The model has a MAE of approximately 72186.35. This value indicates, on average, how far off the predictions are from the actual house prices.  1. MSE(Mean Squared Error):  * MSE measures the average squared difference between the actual and predicted values. * The model has a MSE of approximately 8.13 billion. This provides a measure of the average squared “distance” between the predictions and the actual values.  1. RMSE(Root Mean Squared Error):  * RMSE is the square root of the MSE and provides a measure of the average absolute error. * The model has an RMSE of approximately 90193.37x, indicating the average absolute difference between the predicted and actual values.   Summary:   * Simple linear regression deals with one independent variable, providing a straightforward analysis of the relationship between two variables. * Multivariate linear regression extends the analysis to multiple independent variables, allowing for a more comprehensive understanding of how a set of features collectively influences the dependent variable.   In both cases, the goal is to estimate the coefficients that minimize the difference between the predicted values and the actual values (minimize the error term C). The choice between simple and multivariate linear regression depends on the complexity of the relationship you are trying to model and the nature of your dataset. |