Report on Linear Regression Model Training and Pipeline

1. Objective

The primary goal of this project was to build, evaluate, and compare the performance of a Linear Regression model using two different approaches:

1. **Using the `LinearRegression` class from the scikit-learn package.**
2. **Implementing the Linear Regression model from scratch using Gradient Descent.**

The comparison was based on the Mean Squared Error (MSE) and R-squared (R^2) metrics.

2. Dataset

The dataset used is the "Bike Sharing Dataset," which records the hourly count of rental bikes in a city along with various weather and seasonal information.

Key Features:

- **Categorical:** `season`, `weathersit`, `mnth`, `yr`, `hr`, `holiday`, `weekday`, `workingday`, and `day_night` (derived from the `hr` feature).
- **Numerical:** `temp`, `hum`, `windspeed`, `temp_hum`, and `temp_windspeed` (interaction terms).

Target Variable:

- **`cnt`:** The total count of bike rentals during each hour.

3. Data Preprocessing

The preprocessing involved handling both categorical and numerical features:

Numerical Features:

- **Imputation:** Missing values were imputed using the mean.
- **Scaling:** All numerical features were scaled using the `MinMaxScaler` to bring them into the same range.

Categorical Features:

- **Imputation:** Missing values were imputed using the most frequent value.
- **Encoding:** Categorical variables were encoded using `TargetEncoder`, which encodes categories with the mean of the target variable for each category.

4. Model Training

a. Using the `LinearRegression` Package

The Linear Regression model was trained using the `LinearRegression` class from scikit-learn. This model was fitted to the preprocessed training data, and predictions were made on the test data.

b. Implementing Linear Regression from Scratch

The model was implemented using the following steps:

1. **Adding a Bias Term:** An intercept term was added to the features.
2. **Normalization:** Features were normalized using `StandardScaler`.
3. **Gradient Descent:**
    - Weights were initialized to zero and iteratively updated using the gradient descent algorithm to minimize the MSE.
4. **Prediction:** Predictions were made on the test set using the learned weights.

Evaluation Metrics

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values.
- **R-squared (R^2):** Indicates the proportion of the variance in the target variable that is predictable from the features.

Results Comparison

Metric	Using Package	Scratch Implementation
Mean Squared Error (MSE)	**14,974.13**	**49,549.98**
R-squared (R^2)	**0.527**	**-0.565**

Observations:

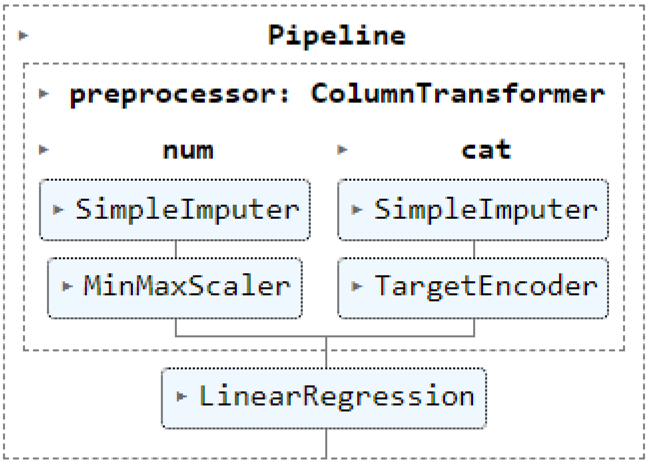
- The model trained using the `LinearRegression` package performed significantly better, with a lower MSE and higher R^2, indicating a better fit.
- The scratch implementation had a higher MSE and a negative R^2, suggesting that the model did not perform well, likely due to challenges in optimizing the gradient descent algorithm.

5. Pipeline Visualization

Below is the visualization of the ML pipeline that combines both preprocessing and model training steps:

Explanation:

- The `preprocessor` step combines numerical and categorical preprocessing.
- The `model` step fits a Linear Regression model to the preprocessed data.



6. Conclusion

This project successfully demonstrated the process of building and evaluating a Linear Regression model using two different approaches. The scikit-learn implementation outperformed the manual implementation, highlighting the importance of using well-optimized libraries for machine learning tasks.