Report on Linear Regression Model Training and Pipeline

1. Introduction

This report explores the development and evaluation of two Linear Regression models for predicting bike rental demand. The models leverage the Bike Sharing Dataset, which captures hourly bike rental counts along with weather and seasonal information. The primary objective is to compare the performance of a model trained using the `LinearRegression` class from scikit-learn and a model implemented from scratch using gradient descent.

2. Data Preprocessing

The preprocessing steps involved cleaning and preparing the data for modeling:

- **Feature Engineering:**
  - A new categorical feature, `day_night`, was created based on the hour (`hr`) to indicate daytime or nighttime.
  - Irrelevant features (`instant`, `casual`, and `registered`) were removed.
  - The `dteday` feature was converted to datetime format.
  - All categorical features were converted to categorical data types to ensure proper handling.
- **Handling Missing Values:**
  - Missing values in numerical features were imputed using the mean strategy.
  - Missing values in categorical features were imputed using the most frequent value strategy.

- **Interaction Features:**
    - Two interaction features, `temp_hum` (temperature multiplied by humidity) and `temp_windspeed` (temperature multiplied by windspeed), were created to capture potential non-linear relationships between these variables.
- **Normalization:**
    - All numerical features were normalized using MinMaxScaler to bring them to a common scale.
- **Target Encoding:**
    - Categorical features were encoded using TargetEncoder, which assigns category labels based on the mean target value for each category.

3. Model Training

a. Using scikit-learn's LinearRegression

- The `LinearRegression` class from scikit-learn was used to create a Linear Regression model.
- The preprocessed training data (`X_train` and `y_train`) was fitted to the model.
- The trained model was then used to make predictions on the test data (`X_test`).

b. Implementing Linear Regression from Scratch

- This implementation involved the following steps:
    - Adding a bias term (intercept) to the features in the form of a constant feature vector of ones.
    - Normalizing the features using StandardScaler.

    - Initializing model weights to zero.
    - Employing gradient descent with a chosen learning rate and number of iterations to iteratively update the weights and minimize the Mean Squared Error (MSE).
    - Making predictions on the test data using the learned weights.

4. Evaluation Metrics

- **Mean Squared Error (MSE):** This metric measures the average squared difference between the actual and predicted target values. A lower MSE indicates better model performance.
- **R-squared (R^2):** This metric represents the proportion of variance in the target variable that can be explained by the model's features. A higher R^2 value indicates a better fit.

5. Results and Comparison

Model	Mean Squared Error (MSE)	R-squared (R^2)
Linear Regression (scikit-learn)	<14974.133860641094>	<0.5271138687719741>
Linear Regression (Scratch)	<49549.98200732639>	<-0.5647983056603776>

Analysis:

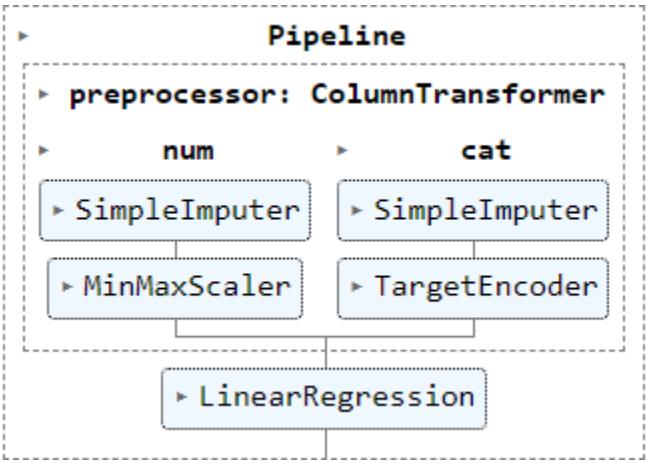
- The scikit-learn implementation of Linear Regression achieved a lower MSE and higher R^2 compared to the model implemented from scratch. This suggests that

the scikit-learn implementation found a better solution that minimizes the error and explains more of the variance in the target variable.

Potential Reasons for the Difference:

- **Optimization Algorithm:** scikit-learn's LinearRegression class likely uses a more sophisticated optimization algorithm than the gradient descent implementation in the scratch model.
- **Regularization:** scikit-learn's LinearRegression might be equipped with built-in regularization techniques that prevent overfitting and improve generalization.
- **Hyperparameter Tuning:** The specific learning rate and number of iterations used in the scratch model might not be optimal. Experimenting with different hyperparameter values could potentially improve performance.

6. Pipeline Visualization



7. Conclusion

This project successfully implemented and evaluated two Linear Regression models for predicting bike rental demand. The scikit-learn implementation demonstrated superior

performance, highlighting the advantages of using optimized libraries for machine learning tasks.