ASSIGNMENT 2

Report on Linear Regression Model Training and Pipeline

-submitted by
BIDISHA SUKAI
(D24CSA002)

Objective:

The Objective of the assignment was to apply the contents of MLOps Lecture 2 to improve the pipeline built and build, evaluate and compare the performance of pipelines based on feature engineering, model selection etc. The tasks assigned were:

1. To create interaction features (at least 2) between numerical variables and justify the choice of features and their impact on performance of the model.
2. To replace OneHotEncoder with TargetEncoder for categorical values
3. Training for Linear Regression Model using two approaches:
    a) LinearRegressor pre-built function from scikit-learn package.
    b) Implementing Linear Regression from scratch and building a class from it (closed form approach)

Dataset:

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

Key Features:

1. **Categorical:**
    - season: The season during which the rental occurred.
    - weathersit: Weather conditions.

- mnth: Month of the year.
- yr: Year of the observation.
- hr: Hour of the day.
- holiday: Whether the day is a holiday.
- weekday: Day of the week.
- workingday: Whether the day is a working day.
- day_night: A derived feature indicating whether it is day or night, based on the hour (hr).

2. **Numerical:**
    - temp: Temperature in Celsius.
    - Atemp: Feels like temperature or apparent temperature
    - hum: Humidity level.
    - windspeed: Wind speed.

3. **Target Variable:**
    - cnt: The total number of bike rentals recorded during each hour.

Our Approach:

1. **Creation of interaction features:**

   Interaction features for numerical and categorical features as well as a mixture of both were created. Even though only the numerical feature interactions were used, here is a brief description of all the interactions created:

   1. **temp_hum**: this is a numerical interaction term. It is used to combine the effects of interaction of temperature and humidity for bike rentals. High humidity on days with high temperature has probability of leading to low bike rentals. The humidity thus affects model performance while combining with temperature.
   2. **temp_windspeed**: this is a numerical interaction term. It is used to combine the effects of interaction of temperature and

windspeed for bike rentals. Higher windspeed lowers temperatures on hotter days thus promoting bike rentals while high winds in cold weather keeps people inside, having a negative impact on bike rentals. Thus Windspeed influences bike rentals significantly.

3. **hum_windspeed**: this is a numerical interaction term. It is used to combine the effects of interaction of humidity and windspeed. Humidity can create varied levels of comfort depending on the windspeed. If the windspeed is low on a high humidity day, it is stifling to be outdoors. So higher windspeeds promote more bike rentals on high humidity days.
4. **weekd_workd**: this is a categorical interaction term. It combines the effects of interaction of whether a weekday is a workingday. It enhances the model performance as bike rental patterns differ if a weekday is a workday or not.
5. **holid_workd**: this is a categorical interaction term. It combines the effects of interaction of holidays with workdays. Model performance is significantly affected if a workday is declared a holiday or if people are called in to work on a holiday. If roads are closed for a holiday people are more likely to use bikes to commute.
6. **season_temp**: this is a mixed interaction term. It combines the effects of season and temperature. Warmer temperatures in the winter season encourages more rentals. While lower temperatures in the summer season encourages more rentals.

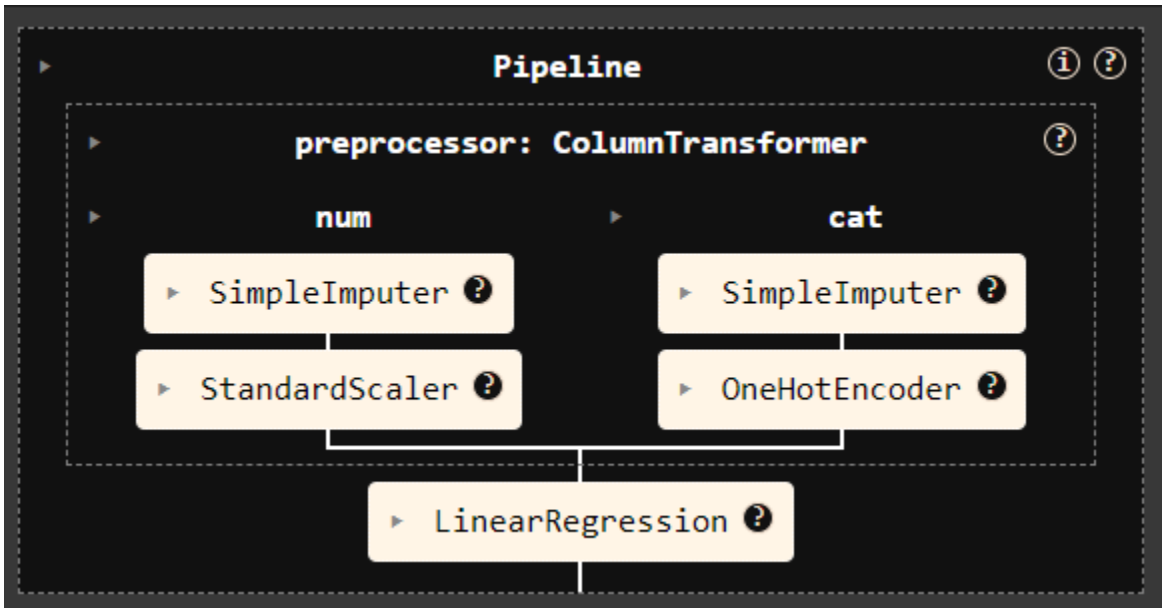
2. Use of Target Encoder:

We had initially used a OneHotEncoder to encode the categorical values, however we now use a TargetEncoder to encode them. One-Hot Encoding converts each categorical variable into a new binary (0 or 1) feature for each unique category. While Target Encoding (also known as Mean Encoding) replaces each category in a categorical variable with the mean of the target variable for that

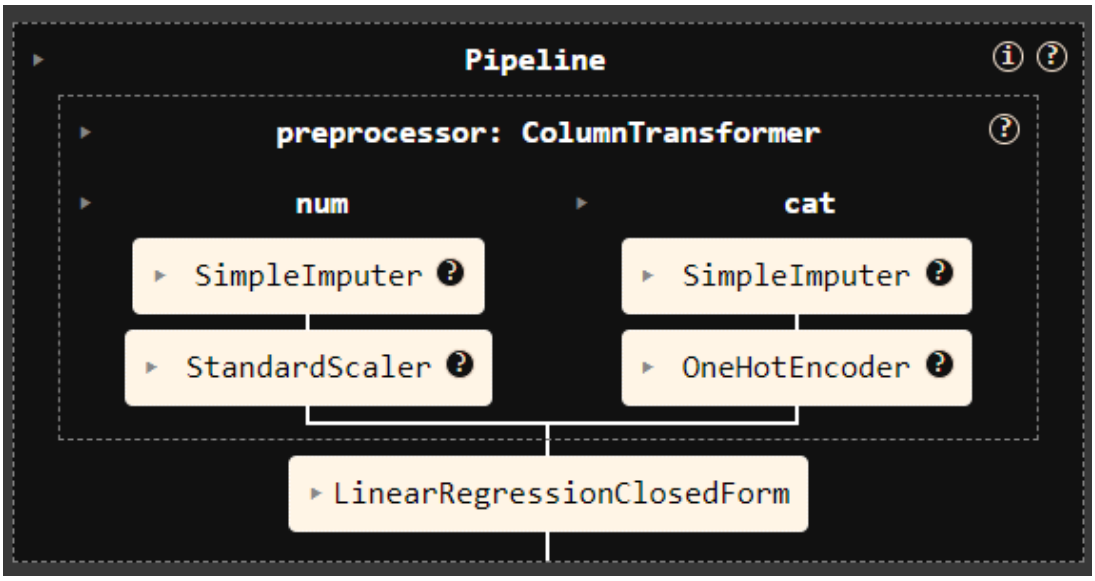
category. This approach creates a single new feature where the value is derived from the relationship between the category and the target variable.

Initial pipeline:

a) For LinearRegression model (scikit):

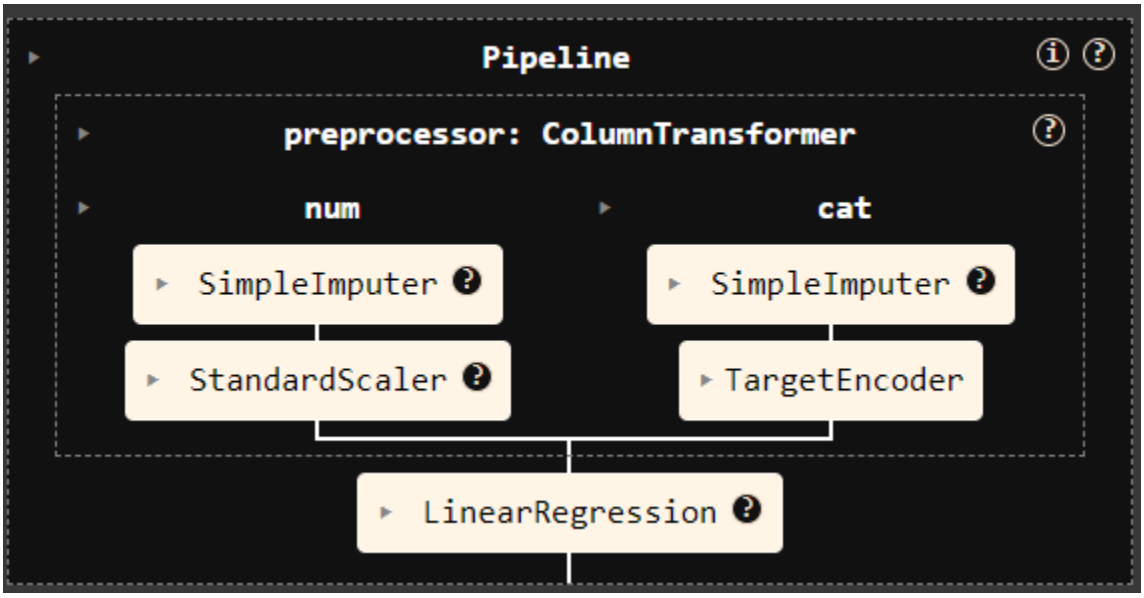


b) For LinearRegression from scratch:

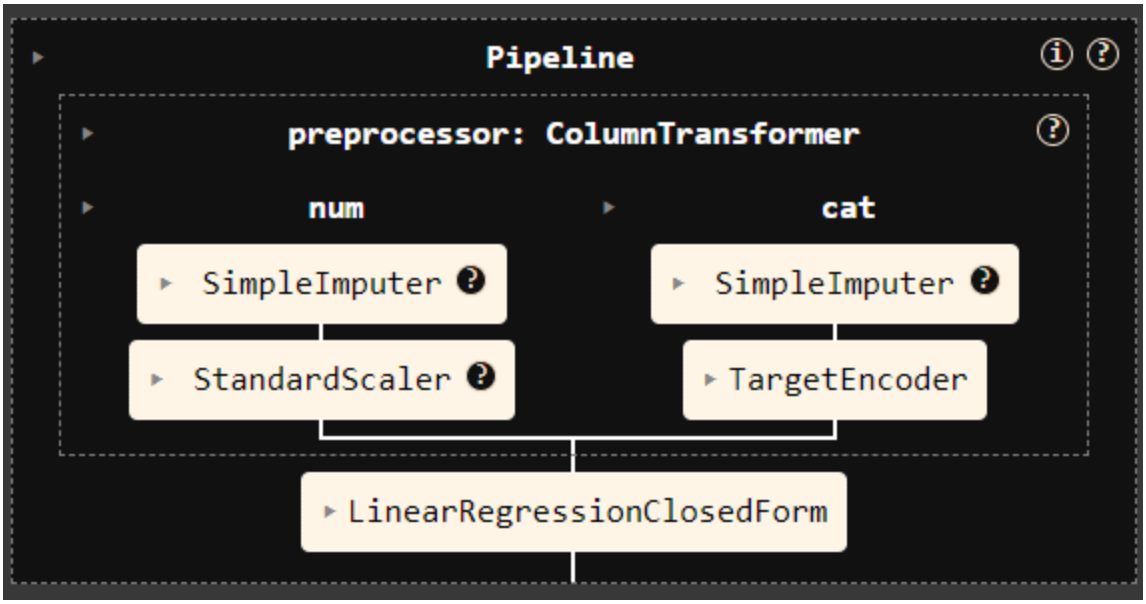


Modified Pipeline:

a) For LinearRegression model (scikit):



b) For LinearRegression from scratch:



3. **Training Linear Regression models:**
   a) **LinearRegressor pre-built function from scikit-learn package:**
   The Linear Regression model, implemented using the `LinearRegression` class from scikit-learn, was trained on the preprocessed training data to learn the relationship between features and the target variable. After training, the model was used to generate predictions on the test data to evaluate its performance.

   b) **Implementing Linear Regression from scratch and building a class from it (closed form approach):**
   The `LinearRegressionClosedForm` class implements a linear regression model using the closed-form solution (normal equation) to compute model parameters.
   In the “fit” method, a column of ones is added to the feature matrix X to account for the intercept term. The closed-form solution is then used to compute the model parameters theta by solving

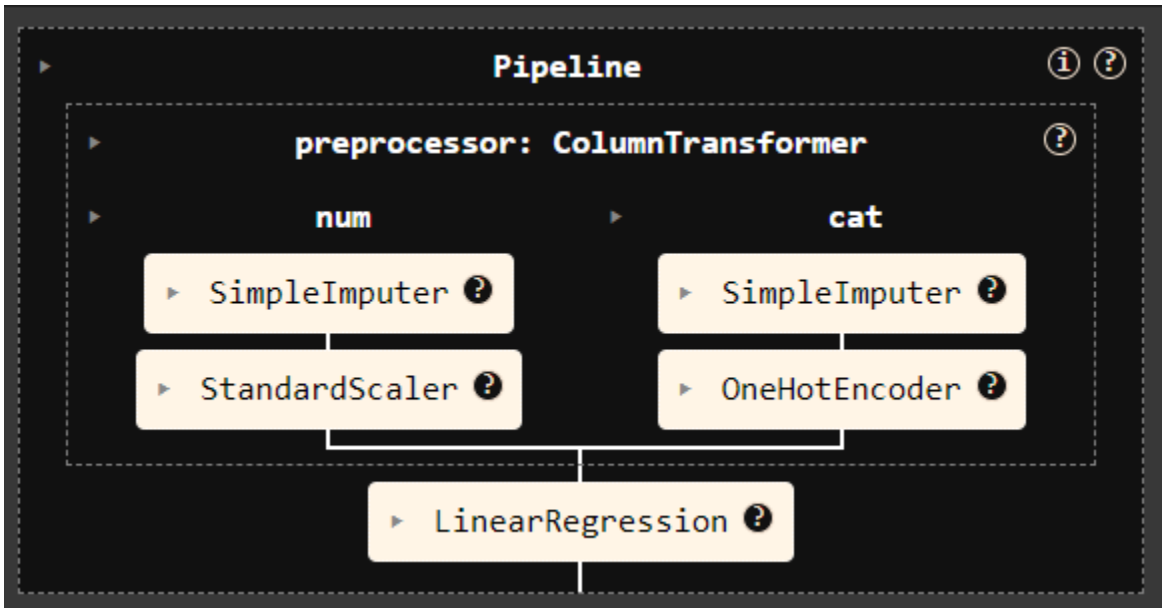
$$\theta = (X^T X)^{-1} X^T y.$$

The predict method adds the intercept term to the input features and uses the computed theta to make predictions by calculating

$$y_{\text{pred}} = X_b \cdot \theta$$

RESULT COMPARISON:

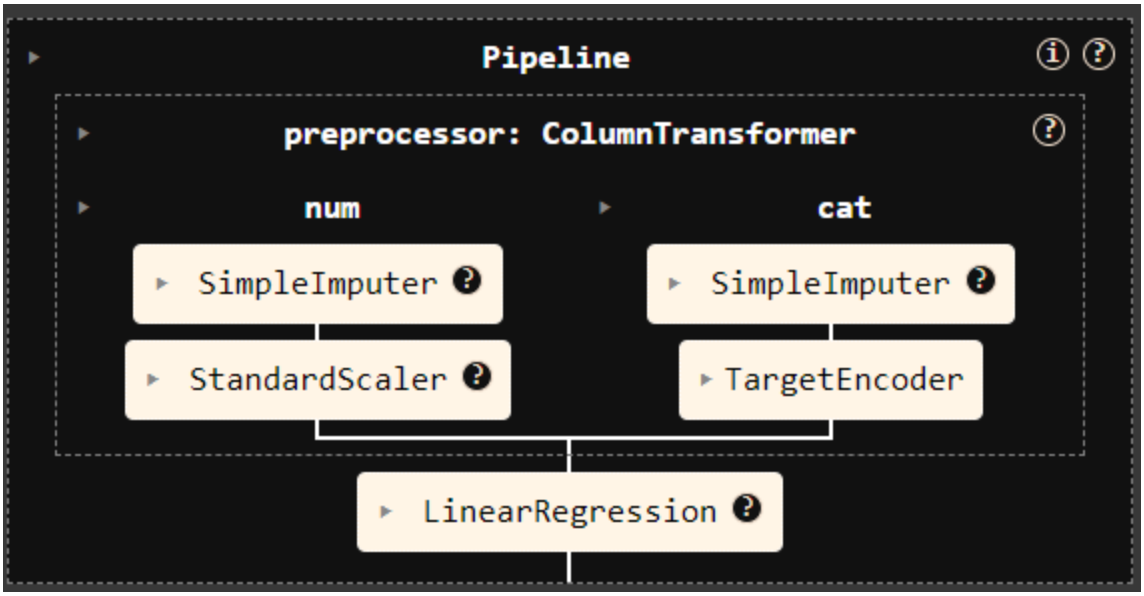
1. LinearRegressor pre-built function from scikit-learn package:



Error values for one hot encoder:

Mean Squared Error: 19472.41596992813

R^2 Score: 0.41391215994488595

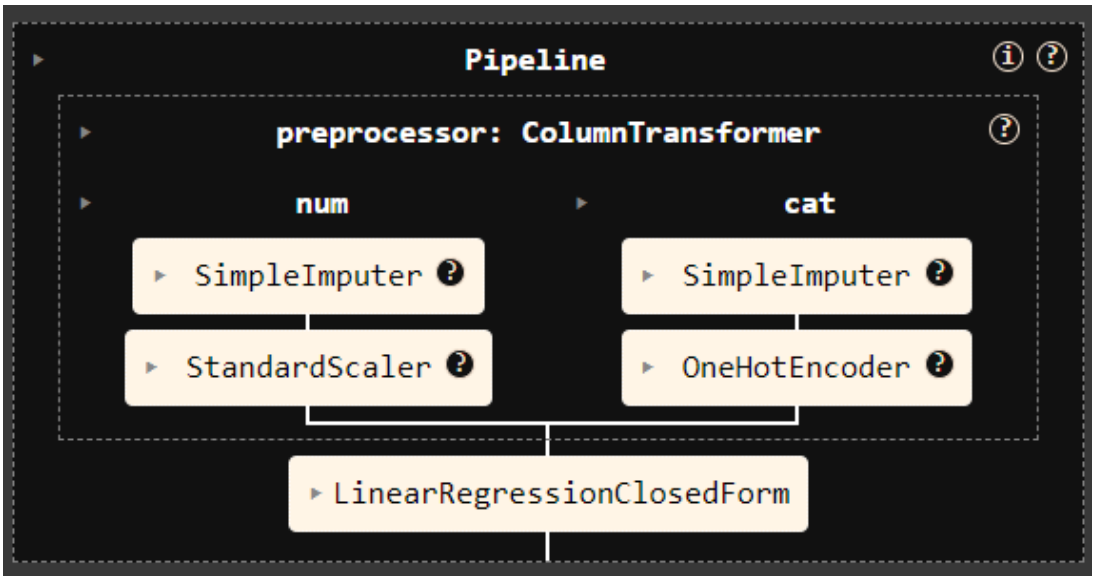


Error values for one target encoder:

Mean Squared Error: 19955.37802991602

R^2 Score: 0.3993757926546633

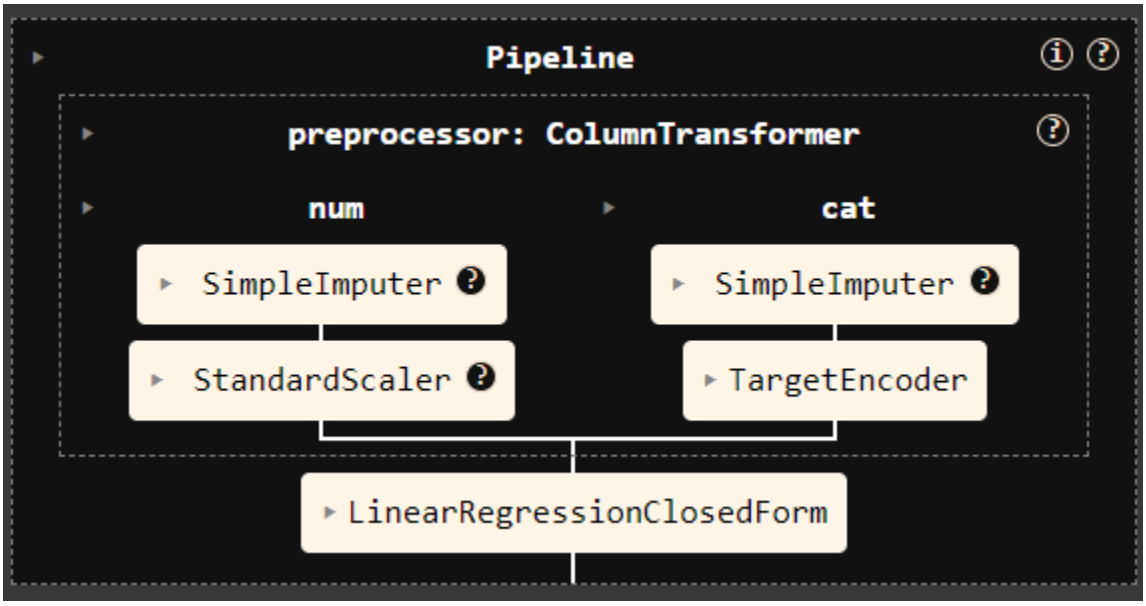
2. Implementing Linear Regression from scratch and building a class from it (closed form approach):



Error values for one hot encoder:

Mean Squared Error: 388879.6786290906

R^2 Score: -10.704641644931533



Error values for target encoder:

Mean Squared Error: 19955.37802991598

R^2 Score: 0.3993757926546645

Results Table:

1. **For OneHotEncoder**

Metric	**Using Package: LinearRegression**	**Scratch Implementation: LinearRegressionClosedForm**
MSE	19472.41596992813	388879.6786290906
R Square	0.41391215994488595	-10.704641644931533

2. **For TargetEncoder**

Metric	**Using Package: LinearRegression**	**Scratch Implementation: LinearRegressionClosedForm**
MSE	19955.37802991602	19955.37802991598
R Square	0.3993757926546633	0.3993757926546645

Conclusion:

We find that for the numerical features we have taken , the TargetEncoder performs slightly lower than the OneHotEncoder for the LinearRegression Model from scikit library. While the OneHotEncoder yields massive error for the Linear Regression from Scratch model, while the TargetEncoder yields the same results for the Linear Regression from Scratch model. This confirms the accuracy of the TargetEncoder's performance across the models indicating successful linear regression implementation. However deterioration of OneHotEncoder results indicates loss of columns along the pipeline even after adding columns of ones.