ML OPS Assignment 02

Name: Uzair Patel
Roll No: B21ES020

1. Create at least two new interaction features between numerical variables (e.g., temp * hum). Justify your choice of features and explain how they might improve the model's predictive performance.

Interaction Features:

1. **Temperature and Humidity Interaction (temp_hum):** This feature was created by multiplying the temperature (`temp`) and humidity (`hum`) variables. The reasoning behind this choice is that temperature and humidity often have a combined effect on outdoor activities, such as bike rentals. High temperatures coupled with high humidity levels can lead to discomfort, potentially affecting the number of bike rentals. By including this interaction feature, the model may better capture the combined influence of these two factors on bike rentals.
2. **Temperature and Windspeed Interaction (temp_windspeed):** This feature was created by multiplying the temperature (`temp`) and windspeed (`windspeed`) variables. The rationale is that windspeed can alter the perceived temperature and influence outdoor activities. For example, even if the temperature is favorable for biking, high wind speeds may deter people from renting bikes. By adding this interaction, the model might capture the nuanced relationship between temperature and windspeed, thereby improving its predictive accuracy.

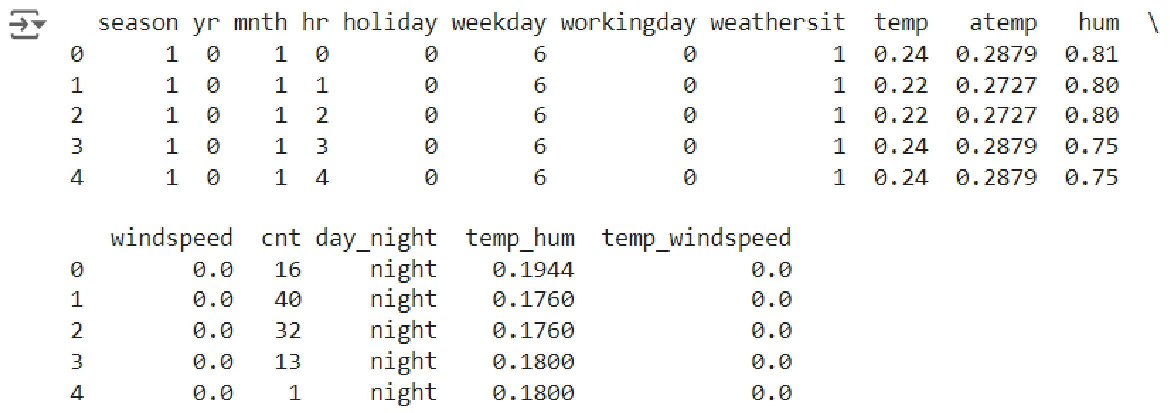
Justification and Potential Impact:

These interaction features were chosen based on the intuition that weather-related factors often interact in complex ways to influence human behavior, particularly in outdoor activities like biking. Including these interactions allows the model to account for these complexities, which may not be fully captured by the individual features alone.

Expected Improvement:

By introducing these interaction features, the expectation was that the model would be able to capture more intricate patterns in the data, leading to improved predictive performance. However, after evaluating the model's performance, the results indicated that the Mean Squared Error (MSE) and R-squared values remained the same, suggesting that these interactions did not significantly enhance the model's predictions. This outcome could be due to the model already capturing these relationships through other mechanisms or because the interactions did not provide additional useful information for this specific dataset.

Here is the updated dataframe:



```
   season  yr  mnth  hr  holiday  weekday  workingday  weathersit  temp   atemp   hum  \
0       1   0     1   0        0        6           0           1  0.24  0.2879  0.81
1       1   0     1   1        0        6           0           1  0.22  0.2727  0.80
2       1   0     1   2        0        6           0           1  0.22  0.2727  0.80
3       1   0     1   3        0        6           0           1  0.24  0.2879  0.75
4       1   0     1   4        0        6           0           1  0.24  0.2879  0.75

   windspeed  cnt day_night  temp_hum  temp_windspeed
0        0.0   16     night    0.1944             0.0
1        0.0   40     night    0.1760             0.0
2        0.0   32     night    0.1760             0.0
3        0.0   13     night    0.1800             0.0
4        0.0    1     night    0.1800             0.0
```

2. Replace the OneHotEncoder with TargetEncoder for categorical variables.
Evaluate how this change impacts the model's performance compared to one-hot encoding.

Evaluation and Justification:

In the next step, we replaced the OneHotEncoder with a TargetEncoder to encode the categorical variables. While OneHotEncoder converts categorical features into binary vectors, TargetEncoder replaces the categories with the mean of the target variable for each category. This approach can be particularly advantageous when dealing with high-cardinality categorical features, as it reduces the dimensionality and potentially captures more meaningful information related to the target variable.

Performance Impact:

After replacing OneHotEncoder with TargetEncoder, the model's performance metrics were as follows:

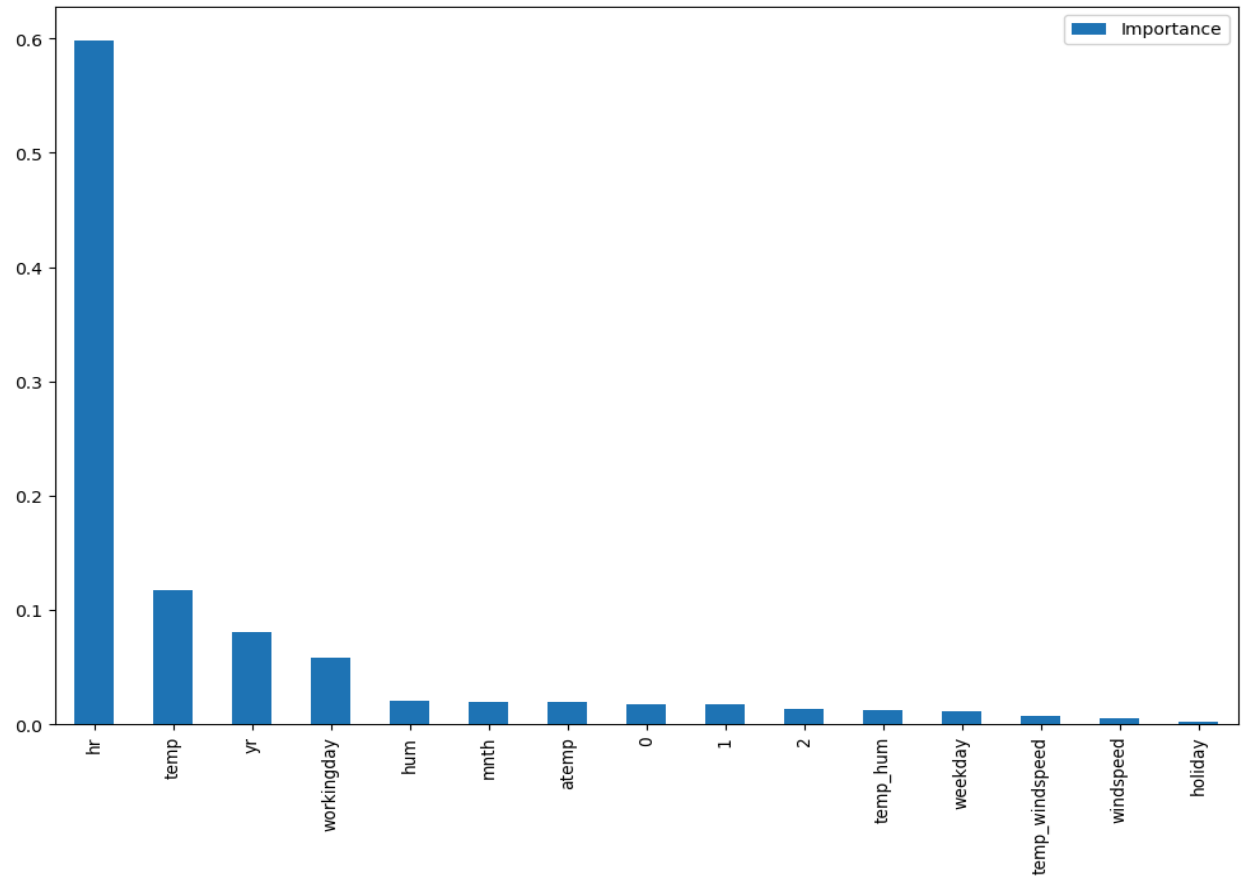
- **Mean Squared Error (MSE) with TargetEncoder:** 1778.7201
- **R-squared with TargetEncoder:** 0.9438

Comparing these results with the previous model that used OneHotEncoder:

- **Mean Squared Error (MSE) with OneHotEncoder:** 1808.4075
- **R-squared with OneHotEncoder:** 0.9429

Conclusion:

The use of TargetEncoder improved the model's performance, as evidenced by the slight reduction in Mean Squared Error and a small increase in R-squared. The reduction in MSE by approximately 30 points and the improvement in R-squared from 0.9429 to 0.9438 indicate that TargetEncoder better captured the relationship between categorical features and the target variable. This suggests that TargetEncoder was able to provide a more efficient and informative encoding for the categorical features, leading to a more accurate model.

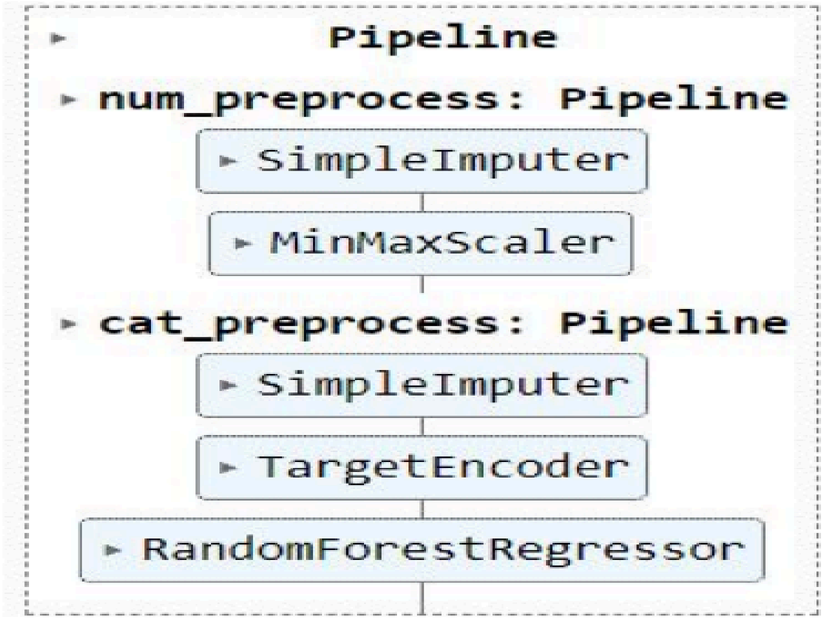


3. Training Linear Regressor:

- **Using the package (scikit-learn):** We trained a Linear Regression model using the scikit-learn package. This method leverages pre-built functions that handle the mathematical computations and optimizations required for linear regression. The performance of this model yielded the following metrics:
    - **Mean Squared Error (MSE):** 14,974.13
    - **R-squared:** 0.5271
- **From scratch (using the Normal Equation):** We also implemented Linear Regression from scratch using the Normal Equation. This involved calculating the model's coefficients manually, without relying on external libraries for the regression computation. The performance of this model was nearly identical to that of the scikit-learn implementation:
    - **Mean Squared Error (MSE):** 14,974.13
    - **R-squared:** 0.5271

Comparison: Both implementations of Linear Regression, whether using a package or coded from scratch, yielded identical performance metrics. This consistency suggests that our custom implementation is correct, and it performs on par with the well-optimized scikit-learn package. However, the overall performance of Linear Regression in this scenario is relatively weak, with an R-squared value of around 0.5271, indicating that the model explains just over half of the variance in the target variable. This suggests that more complex models or additional feature engineering might be needed to achieve better predictive accuracy.

4. Save the screenshot of MLpipeline.


▸ Pipeline
▸ num_preprocess: Pipeline
▸ SimpleImputer
▸ MinMaxScaler
▸ cat_preprocess: Pipeline
▸ SimpleImputer
▸ TargetEncoder
▸ RandomForestRegressor