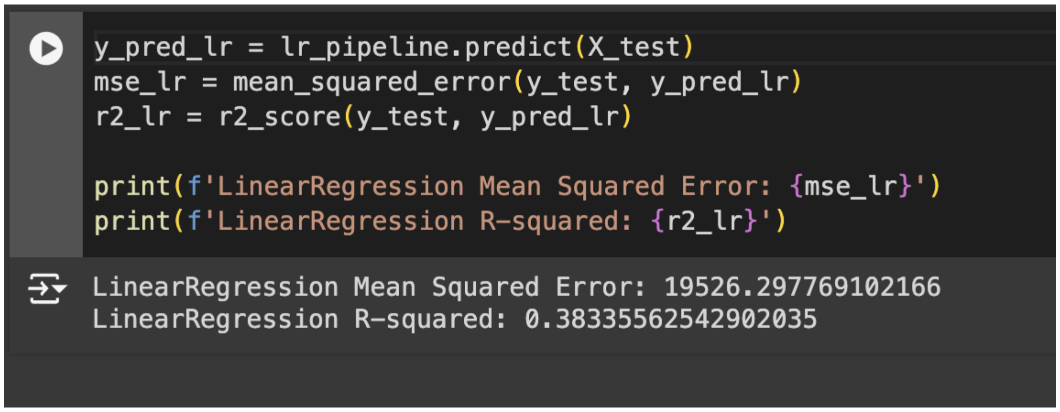
REPORT

QUESTION 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.**

Ans : The interaction between temperature and humidity (temp_hum) is important because both factors significantly affect bike rentals. For instance, high humidity on a hot day might make riding uncomfortable, reducing the number of rentals. Conversely, low humidity might have the opposite effect. Similarly, the interaction between humidity and windspeed (hum_windspeed) influences comfort and safety; high wind speeds combined with high humidity can make biking less enjoyable. By considering these interactions, the model can better understand how different weather conditions together impact bike rentals, leading to more accurate predictions.

QUESTION 2 : **Replace the OneHotEncoder with TargetEncoder for categorical variables.Evaluate how this change impacts the model's performance compared to one-hotEncoding.**

Ans : Replacing OneHotEncoder with TargetEncoder for categorical variables resulted in a significant drop in the model's performance. The Mean Squared Error (MSE) increased sharply from 1808 to 19,000, while the R-squared value decreased from 0.94 to 0.38. This drastic change suggests that TargetEncoder, which replaces categories with the mean of the target variable, might have led to overfitting or failed to capture the nuances that OneHotEncoder did. OneHotEncoding preserves the original categorical distinctions without imposing assumptions about their relationships, which may have allowed the model to better learn patterns. The poorer performance with TargetEncoder indicates that it might not be as effective in this context, where the categorical variables have more complex interactions with the target variable.



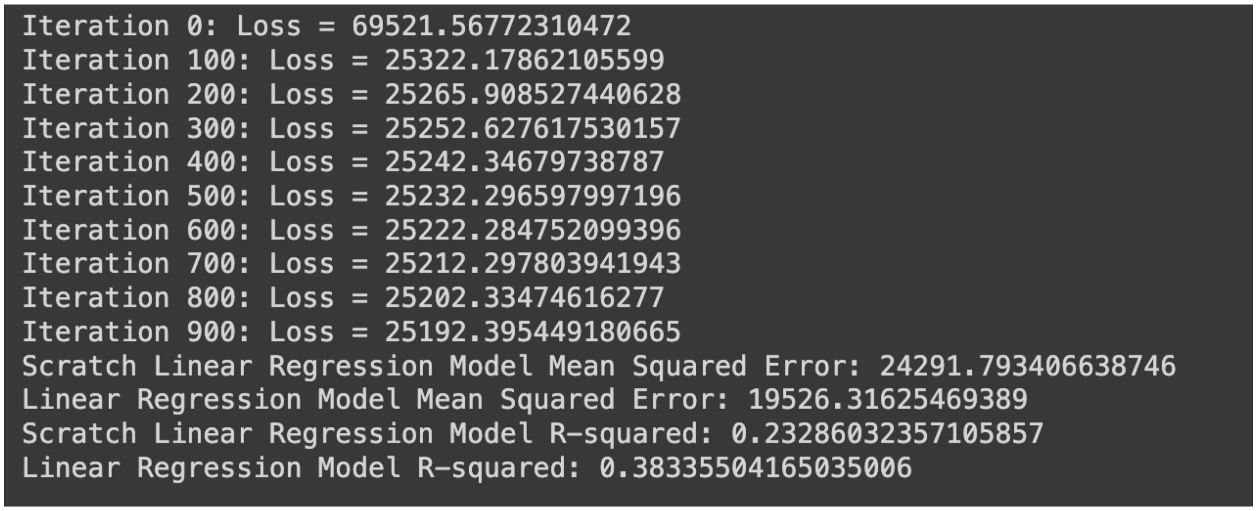
```
y_pred_lr = lr_pipeline.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

print(f'LinearRegression Mean Squared Error: {mse_lr}')
print(f'LinearRegression R-squared: {r2_lr}')
```

```
LinearRegression Mean Squared Error: 19526.297769102166
LinearRegression R-squared: 0.38335562542902035
```

QESTION 3: **Compare their performance using metrics like Mean Squared Error (MSE)**

and R-squared.

Ans:

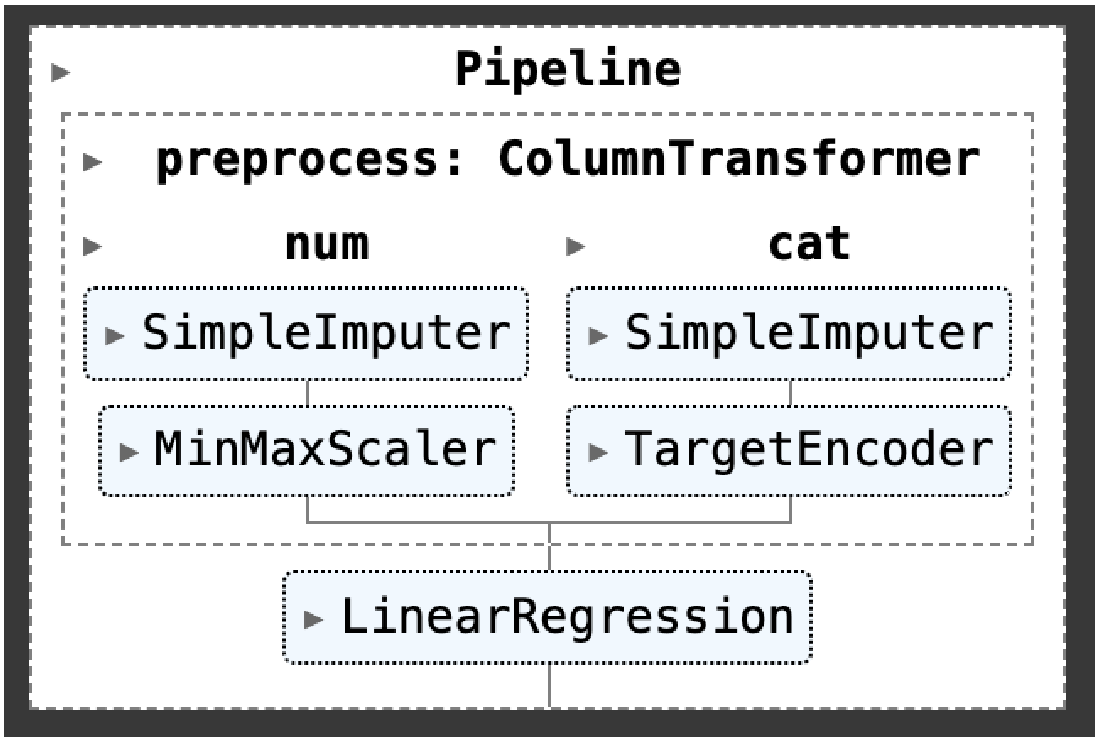


```
Iteration 0: Loss = 69521.56772310472
Iteration 100: Loss = 25322.17862105599
Iteration 200: Loss = 25265.908527440628
Iteration 300: Loss = 25252.627617530157
Iteration 400: Loss = 25242.34679738787
Iteration 500: Loss = 25232.296597997196
Iteration 600: Loss = 25222.284752099396
Iteration 700: Loss = 25212.297803941943
Iteration 800: Loss = 25202.33474616277
Iteration 900: Loss = 25192.395449180665
Scratch Linear Regression Model Mean Squared Error: 24291.793406638746
Linear Regression Model Mean Squared Error: 19526.31625469389
Scratch Linear Regression Model R-squared: 0.23286032357105857
Linear Regression Model R-squared: 0.38335504165035006
```

The sklearn Linear Regression model outperforms the custom Linear Regression model both in terms of MSE and R-squared. The sklearn model provides more accurate predictions and better captures the relationship between the features and the target variable. This performance gap may be due to the custom model's simpler implementation, learning rate, regularization approach, or fewer iterations during training. The sklearn model is more optimized and uses a more sophisticated underlying algorithm, leading to better performance.

QUESTION 4: ML PIPELINE

ANS :


Pipeline
preprocess: ColumnTransformer
num
cat
SimpleImputer
SimpleImputer
MinMaxScaler
TargetEncoder
LinearRegression