MLops Assignment-2 Report

Assignment: Enhancing and Optimizing an MLOps Pipeline

1) New Interaction Features:

Let us first see the original features as mentioned in the dataset.

- **instant**: record index
- **dteday** : date
- **season** : season (1:springer, 2:summer, 3:fall, 4:winter)
- **yr** : year (0: 2011, 1:2012)
- **mnth** : month (1 to 12)
- **hr** : hour (0 to 23)
- **holiday** : weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
- **weekday** : day of the week
- **workingday** : if day is neither weekend nor holiday is 1, otherwise is 0.
- **weathersit** :
  - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- **temp** : Normalized temperature in Celsius. The values are divided to 41 (max)
- **atemp**: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- **hum**: Normalized humidity. The values are divided to 100 (max)

- **windspeed**: Normalized wind speed. The values are divided to 67 (max)
- **casual**: count of casual users
- **registered**: count of registered users

- After applying Random Forest Regressor model on the above dataset, it is observed that the most important features are hr > temp > yr > workingday > hum > atemp > mnth in decreasing order of importance.
- **New Features Addition:**
  1) **hr*temp**
     This feature tells us about the environmental temperature during different times of the day that might affect bike rental patterns. For example - temperatures during midday are high and that might deter people from renting bikes where temperatures during evening might promote more bike rentals.
  2) **temp*windspeed**
     This feature tells us about interaction between environmental temperature and wind speed which can potentially influence bike usage and thus bike rentals. For example - when temperatures are high, high wind speed might feel more comfortable than when temperatures are high with no/low wind speed.
  3) **hr*workingday**
     This feature tells us about the interaction between the hour of the day and whether it's a working day or not. This feature can capture different patterns in bike usage. For example - 8:30 AM - 10:30 AM of the working days are peak commuting hours for the working class and using this feature we can extract this information.

2) Comparison between OneHotEncoder and TargetEncoder

OneHotEncoder: Converts categorical variables into a binary matrix, where each category is represented as a separate binary column. It produces a binary matrix where each category is represented by a column, and each row corresponds to an observation. If the observation belongs to a particular category, the value is 1; otherwise, it's `0`.

TargetEncoder: Replaces each category with the mean of the target variable for that category. It outputs a numerical representation of the categorical feature, where each category is replaced by a summary statistic (usually the mean) of the target variable for that category.

Using RandomForestRegressor model the mean squared error(MSE) and R-squared(R^2) values for both encoders are as follows:

a) OneHotEncoder:
   i) MSE = 1808.4074990292243
   ii) R^2 = 0.9428901308176855

b) TargetEncoder:
   i) MSE = 1734.5910494106192
   ii) R^2 = 0.9452212690061106

Performance Comparison:

- **Mean Squared Error (MSE):**
  - **OneHotEncoder MSE:** 1808.41
  - **TargetEncoder MSE:** 1734.59
  - **Conclusion:** The model with TargetEncoder has a lower MSE compared to OneHotEncoder, indicating that, on average, the predictions from the model using TargetEncoder are closer to the actual values. This suggests that TargetEncoder provides a slightly better fit to the data in terms of minimizing prediction errors.

- **R-squared (R^2):**
  - **OneHotEncoder R^2:** 0.9429
  - **TargetEncoder R^2:** 0.9452
  - **Conclusion:** The R^2 value is slightly higher for the model using TargetEncoder, meaning it explains a slightly greater proportion of the variance in the target variable. This indicates that the model using TargetEncoder has a marginally better overall fit to the data.

3) Comparison between LinearRegressor Package and LinearRegressor from scratch:

a) LinearRegressor Package

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
```

i) MSE = 14255.542995183217

ii) R^2 = 0.549807111497375

b) LinearRegressor from scratch

```
import numpy as np
def tranp(A):
  rows, cols = A.shape
  c = np.zeros((cols,rows))
  for i in range(rows):
    for j in range(cols):
      c[j][i] = A[i][j]
  return c

class Lin_Reg():
    def __init__(self):
      self.coefficient = None

    def fit(self, X, y):
      rows, cols = X.shape
      X = np.column_stack((np.ones(rows), X))
      Xtrans = tranp(X)
      ma = np.linalg.solve(np.dot(Xtrans, X), np.dot(Xtrans,
y))
      self.coefficient = ma

    def pred(self, X):
      rows, cols = X.shape
      X = np.column_stack((np.ones(rows), X))
      return np.dot(X, self.coefficient)
```

```
lr = Lin_Reg()
lr.fit(X_train, y_train)
```

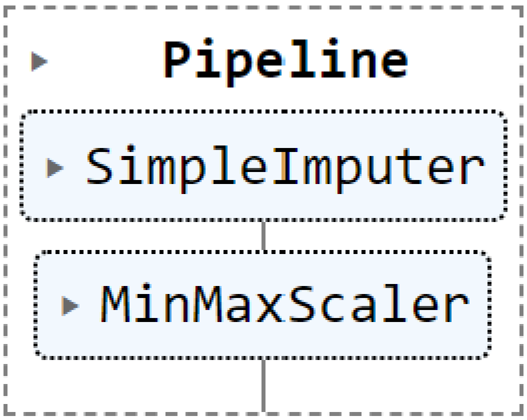
i) MSE = 14255.54299518321

ii) R^2 = 0.5498071114973753

4) Pipelines:

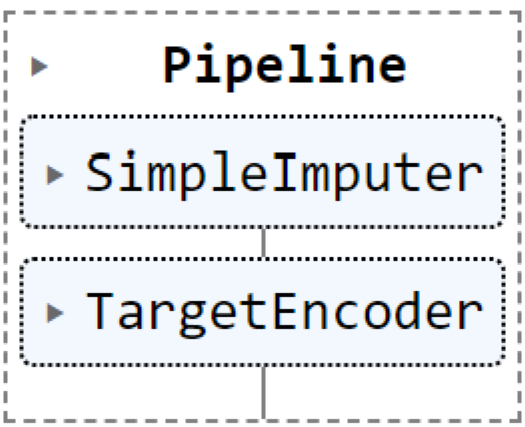
a) Numerical Pipeline

```
numerical_pipeline
```



b) Categorical Pipeline

```
categorical_pipeline
```



c) Final Pipeline