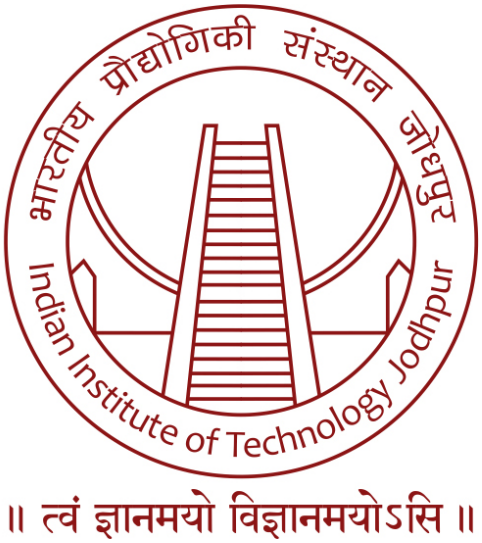
CSP7040 : MLOps

Lab Report



Name:	**Mohammad Haris Ansari**
Roll Number:	**M23IQT003**
Program:	**MTech Quantum Technology**

Chapter 1

Assignment-2

1.1 Objective

- Create new interaction features between numerical variables.
- Replace OneHotEncoder with TargetEncoder for categorical variables and evaluate the impact.
- Train a linear regression using both a package and from scratch, and compare their performance.
- Save a screenshot of the MLOps pipeline.

1.2 Problem 1(a): New Features Creation

In this part of the assignment, we created new interaction features to potentially improve the predictive performance of the model.

Listing 1.1: Creating Interaction Features

```
# Creating new interaction features
df['temp_hum'] = df['temp'] * df['hum']
df['temp_windspeed'] = df['temp'] * df['windspeed']
```

Interaction Features:

(Temperature * Humidity): *Justification:* Temperature and humidity interact to influence outdoor behavior. High temperatures and humidity may reduce outdoor activities, affecting bike rentals.

(Temperature * Windspeed): *Justification:* Wind and temperature affect bike usage. Low temperatures and high winds discourage biking.

1.3 Problem 1(b): Replace OneHotEncoder with TargetEncoder

Here, we replaced the OneHotEncoder with a TargetEncoder to assess its impact on model performance.

Listing 1.2: Creating Interaction Features

```
categorical_pipeline = Pipeline([
('imputer', SimpleImputer(strategy='most_frequent')),
('target', TargetEncoder())
])
# Transforming above
X_encoded = categorical_pipeline.fit_transform(X[
    ↪ categorical_features],y)
# Converting it to a dataframe

X_encoded = pd.DataFrame(X_encoded,
columns=categorical_pipeline.named_steps['target'].
    ↪ get_feature_names_out(categorical_features))
# Encoded categorical features + Numerical features
X = pd.concat([X.drop(columns=categorical_features), X_encoded],
    ↪ axis=1)

X.columns = X.columns.astype(str)
```

1.3.1 Result Comparison: OneHotEncoder vs TargetEncoder

The results of the comparison between OneHotEncoder and TargetEncoder show that TargetEncoder slightly outperforms OneHotEncoder in terms of both Mean Squared Error (MSE) and R-squared:

OneHotEncoder Performance:

- Mean Squared Error: 1838.4677
- R-squared: 0.9419

TargetEncoder Performance:

- Mean Squared Error: 1778.4512
- R-squared: 0.9438

1.4 Problem 1(c): Train Linear Regressor using Package

We trained a linear regressor using the ‘LinearRegression‘ package from ‘scikit-learn‘.

Listing 1.3: Creating Interaction Features

```
from sklearn.linear_model import LinearRegression
```

```
sklearn_model = LinearRegression()

sklearn_model.fit(X_train, y_train)

y_pred_sklearn = sklearn_model.predict(X_test)

mse_sklearn = mean_squared_error(y_test, y_pred_sklearn)
r2_sklearn = r2_score(y_test, y_pred_sklearn)

print(f"Sklearn Linear Regression - MSE: {mse_sklearn}, R : {
    r2_sklearn}")
```

1.5 Problem 1(d): Train Linear Regressor from Scratch

This subpart involved implementing a linear regressor from scratch using gradient descent.

```
X_train_aug = np.concatenate([np.ones((X_train.shape[0], 1)),
    X_train], axis=1)
X_test_aug = np.concatenate([np.ones((X_test.shape[0], 1)),
    X_test], axis=1)

optimal_theta = np.linalg.inv(X_train_aug.T @ X_train_aug) @
    X_train_aug.T @ y_train

predictions = X_test_aug @ optimal_theta

mse_manual = mean_squared_error(y_test, predictions)
r2_manual = r2_score(y_test, predictions)

print(f"Custom Linear Regression - MSE: {mse_manual}, R : {
    r2_manual}")
```

1.5.1 Result Comparison: Sklearn vs Scratch Linear Regression

The results for both Sklearn's Linear Regression and the custom implementation show nearly identical performance:

Sklearn Linear Regression:

- Mean Squared Error: 14974.1338
- R-squared: 5271

Linear Regression from Scratch:

- Mean Squared Error: 14974.1338
- R-squared: 5271

1.6 Problem 1(e): Save the Screenshot of ML Pipeline

In this section, the problem is to save and document the machine learning pipeline by taking a screenshot.