21162101012_CBA_Yash_Lakhtariya

ML Practical 3 and 4

Aim - Implement linear regression for given dataset and find model which has highest r2 score and minimum MSE

[32]
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Import datasets

[33]
```python
import pandas as pd
import numpy as np

# Load Bike Data
path_bike = "/content/drive/MyDrive/Colab Notebooks/ML_7/p3_bikedata_daywise.csv"
df_bike = pd.read_csv(path_bike, header=0)

# Load Metro Data
path_metro = "/content/drive/MyDrive/Colab Notebooks/ML_7/p3_metro.csv"
df_metro = pd.read_csv(path_metro, header=0)

# Load Autos Data
path_autos = "/content/drive/MyDrive/Colab Notebooks/ML_7/p3_autos.csv"
df_autos = pd.read_csv(path_autos, header=0)

# Display the first few rows of each DataFrame to verify the loading process
print("Bike Data:")
print(df_bike.head(), "\n")

print("Metro Data:")
print(df_metro.head(), "\n")

print("Autos Data:")
print(df_autos.head(), "\n")
```

```
Bike Data:
   instant      dteday  season  yr  mnth  holiday  weekday  workingday  \
0        1  2011-01-01       1   0     1        0        6           0
1        2  2011-01-02       1   0     1        0        0           0
2        3  2011-01-03       1   0     1        0        1           1
3        4  2011-01-04       1   0     1        0        2           1
4        5  2011-01-05       1   0     1        0        3           1

   weathersit      temp     atemp       hum  windspeed  casual  registered  \
0           2  0.344167  0.363625  0.805833   0.160446     331         654
1           2  0.363478  0.353739  0.696087   0.248539     131         670
2           1  0.196364  0.189405  0.437273   0.248309     120        1229
3           1  0.200000  0.212122  0.590435   0.160296     108        1454
4           1  0.226957  0.229270  0.436957   0.186900      82        1518

    cnt
0   985
1   801
2  1349
3  1562
4  1600

Metro Data:
  holiday    temp  rain_1h  snow_1h  clouds_all weather_main  \
0     NaN  288.28      0.0      0.0          40       Clouds
1     NaN  289.36      0.0      0.0          75       Clouds
2     NaN  289.58      0.0      0.0          90       Clouds
3     NaN  290.13      0.0      0.0          90       Clouds
4     NaN  291.14      0.0      0.0          75       Clouds

  weather_description            date_time  traffic_volume
0    scattered clouds  2012-10-02 09:00:00            5545
1       broken clouds  2012-10-02 10:00:00            4516
2     overcast clouds  2012-10-02 11:00:00            4767
3     overcast clouds  2012-10-02 12:00:00            5026
4       broken clouds  2012-10-02 13:00:00            4918

Autos Data:
   normalized-losses         make fuel-type aspiration num-of-doors  \
0                NaN  alfa-romero       gas        std          two
1                NaN  alfa-romero       gas        std          two
2                NaN  alfa-romero       gas        std          two
3              164.0         audi       gas        std         four
4              164.0         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  length  ...  \
0  convertible          rwd           front        88.6   168.8  ...
1  convertible          rwd           front        88.6   168.8  ...
2    hatchback          rwd           front        94.5   171.2  ...
3        sedan          fwd           front        99.8   176.6  ...
4        sedan          4wd           front        99.4   176.6  ...

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0      111.0    5000.0       21
1         mpfi  3.47    2.68               9.0      111.0    5000.0       21
2         mpfi  2.68    3.47               9.0      154.0    5000.0       19
3         mpfi  3.19    3.40              10.0      102.0    5500.0       24
4         mpfi  3.19    3.40               8.0      115.0    5500.0       18
```

Clean Data

```python
def clean_data(df):
    # Convert date columns to datetime if present
    if 'dteday' in df.columns:
        df['dteday'] = pd.to_datetime(df['dteday'])
    if 'date_time' in df.columns:
        df['date_time'] = pd.to_datetime(df['date_time'])

    # Replace missing values with column means for numeric columns
    numeric_columns = df.select_dtypes(include=[np.number]).columns
    df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

    # Replace missing values in categorical columns with the mode
    categorical_columns = df.select_dtypes(include=['object']).columns
    for col in categorical_columns:
        df[col].fillna(df[col].mode()[0], inplace=True)

    return df

# Clean each dataset
df_bike_clean = clean_data(df_bike)
df_metro_clean = clean_data(df_metro)
df_autos_clean = clean_data(df_autos)

# Display the first few rows of each cleaned DataFrame to verify the cleaning process
print("Cleaned Bike Data:")
print(df_bike_clean.head(), "\n")

print("Cleaned Metro Data:")
print(df_metro_clean.head(), "\n")

print("Cleaned Autos Data:")
print(df_autos_clean.head(), "\n")
```

```
1           2  0.363478  0.353739  0.696087   0.248539     131         670
2           1  0.196364  0.189405  0.437273   0.248309     120        1229
3           1  0.200000  0.212122  0.590435   0.160296     108        1454
4           1  0.226957  0.229270  0.436957   0.186900      82        1518

    cnt
0   985
1   801
2  1349
3  1562
4  1600

Cleaned Metro Data:
     holiday    temp  rain_1h  snow_1h  clouds_all weather_main  \
0  Labor Day  288.28      0.0      0.0          40       Clouds
1  Labor Day  289.36      0.0      0.0          75       Clouds
2  Labor Day  289.58      0.0      0.0          90       Clouds
3  Labor Day  290.13      0.0      0.0          90       Clouds
4  Labor Day  291.14      0.0      0.0          75       Clouds

  weather_description           date_time  traffic_volume
0    scattered clouds 2012-10-02 09:00:00            5545
1       broken clouds 2012-10-02 10:00:00            4516
2     overcast clouds 2012-10-02 11:00:00            4767
3     overcast clouds 2012-10-02 12:00:00            5026
4       broken clouds 2012-10-02 13:00:00            4918

Cleaned Autos Data:
   normalized-losses         make fuel-type aspiration num-of-doors  \
0              122.0  alfa-romero       gas        std          two
1              122.0  alfa-romero       gas        std          two
2              122.0  alfa-romero       gas        std          two
3              164.0         audi       gas        std         four
4              164.0         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  length  ...  \
0  convertible          rwd           front        88.6   168.8  ...
1  convertible          rwd           front        88.6   168.8  ...
2    hatchback          rwd           front        94.5   171.2  ...
3        sedan          fwd           front        99.8   176.6  ...
4        sedan          4wd           front        99.4   176.6  ...

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0      111.0    5000.0       21
1         mpfi  3.47    2.68               9.0      111.0    5000.0       21
2         mpfi  2.68    3.47               9.0      154.0    5000.0       19
3         mpfi  3.19    3.40              10.0      102.0    5500.0       24
4         mpfi  3.19    3.40               8.0      115.0    5500.0       18

   highway-mpg    price  symboling
0           27  13495.0          3
1           27  16500.0          3
2           26  16500.0          1
3           30  13950.0          2
4           22  17450.0          2

[5 rows x 26 columns]
```

LinearRegression-Model for Bike Data

```python
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Prepare the data for regression
X_bike = df_bike_clean[['temp', 'atemp', 'hum', 'windspeed']]  # Features selected
y_bike = df_bike_clean['cnt']  # Target variable

# Split the data into training and testing sets
X_train_bike, X_test_bike, y_train_bike, y_test_bike = train_test_split(X_bike, y_bike, test_size=0.2, random_state=42)

lr_bike = LinearRegression()

# Train the model on the training data
lr_bike.fit(X_train_bike, y_train_bike)

# Make predictions on the test data
y_pred_bike = lr_bike.predict(X_test_bike)

# Evaluate the model using R² score
r2_bike = r2_score(y_test_bike, y_pred_bike)
print(f"Linear Regression R² Score for Bike Dataset: {r2_bike:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_bike = cross_val_score(lr_bike, X_bike, y_bike, cv=5)
cv_mean_bike = np.mean(cv_scores_bike)
print(f"Linear Regression Cross-Validation Score for Bike Dataset: {cv_mean_bike:.4f}")
```

```
Linear Regression R² Score for Bike Dataset: 0.4995
Linear Regression Cross-Validation Score for Bike Dataset: -1.9649
```

Ridge-Model on Bike data

```python
from sklearn.linear_model import Ridge

ridge_bike = Ridge(alpha=1.0)

# Train the model on the training data
ridge_bike.fit(X_train_bike, y_train_bike)

# Make predictions on the test data
y_pred_ridge_bike = ridge_bike.predict(X_test_bike)

# Evaluate the model using R² score
r2_ridge_bike = r2_score(y_test_bike, y_pred_ridge_bike)
print(f"Ridge Regression R² Score for Bike Dataset: {r2_ridge_bike:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_ridge_bike = cross_val_score(ridge_bike, X_bike, y_bike, cv=5)
cv_mean_ridge_bike = np.mean(cv_scores_ridge_bike)
print(f"Ridge Regression Cross-Validation Score for Bike Dataset: {cv_mean_ridge_bike:.4f}")
```

```
Ridge Regression R² Score for Bike Dataset: 0.4869
Ridge Regression Cross-Validation Score for Bike Dataset: -1.8712
```

Lasso-Model on Bike data

```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso

# Standardize the features
scaler = StandardScaler()
X_bike_scaled = scaler.fit_transform(X_bike)

# Split the scaled data into training and testing sets
X_train_bike_scaled, X_test_bike_scaled, y_train_bike, y_test_bike = train_test_split(X_bike_scaled, y_bike, test_size=0.2, random_state=42)

# Initialize Lasso with increased max_iter
lasso_bike = Lasso(alpha=0.1, max_iter=10000)

# Train the model on the scaled training data
lasso_bike.fit(X_train_bike_scaled, y_train_bike)

# Make predictions on the scaled test data
y_pred_lasso_bike = lasso_bike.predict(X_test_bike_scaled)

# Evaluate the model using R² score
r2_lasso_bike = r2_score(y_test_bike, y_pred_lasso_bike)
print(f"Lasso Regression R² Score for Bike Dataset (Scaled): {r2_lasso_bike:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_lasso_bike = cross_val_score(lasso_bike, X_bike_scaled, y_bike, cv=5)
cv_mean_lasso_bike = np.mean(cv_scores_lasso_bike)
print(f"Lasso Regression Cross-Validation Score for Bike Dataset (Scaled): {cv_mean_lasso_bike:.4f}")
```

```
Lasso Regression R² Score for Bike Dataset (Scaled): 0.4994
Lasso Regression Cross-Validation Score for Bike Dataset (Scaled): -1.9648
```

LinearRegression-Model on Metro Data

```python
# Prepare the data for regression
X_metro = df_metro_clean[['temp', 'rain_1h', 'snow_1h', 'clouds_all']]  # Features selected
y_metro = df_metro_clean['traffic_volume']  # Target variable

# Split the data into training and testing sets
X_train_metro, X_test_metro, y_train_metro, y_test_metro = train_test_split(X_metro, y_metro, test_size=0.2, random_state=42)

lr_metro = LinearRegression()

# Train the model on the training data
lr_metro.fit(X_train_metro, y_train_metro)

# Make predictions on the test data
y_pred_metro = lr_metro.predict(X_test_metro)

# Evaluate the model using R² score
r2_metro = r2_score(y_test_metro, y_pred_metro)
print(f"Linear Regression R² Score for Metro Dataset: {r2_metro:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_metro = cross_val_score(lr_metro, X_metro, y_metro, cv=5)
cv_mean_metro = np.mean(cv_scores_metro)
print(f"Linear Regression Cross-Validation Score for Metro Dataset: {cv_mean_metro:.4f}")
```

```
Linear Regression R² Score for Metro Dataset: 0.0234
Linear Regression Cross-Validation Score for Metro Dataset: -2.5345
```

Ridge-Model on Metro data

```python
ridge_metro = Ridge(alpha=1.0)

# Train the model on the training data
ridge_metro.fit(X_train_metro, y_train_metro)

# Make predictions on the test data
y_pred_ridge_metro = ridge_metro.predict(X_test_metro)

# Evaluate the model using R² score
r2_ridge_metro = r2_score(y_test_metro, y_pred_ridge_metro)
print(f"Ridge Regression R² Score for Metro Dataset: {r2_ridge_metro:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_ridge_metro = cross_val_score(ridge_metro, X_metro, y_metro, cv=5)
cv_mean_ridge_metro = np.mean(cv_scores_ridge_metro)
print(f"Ridge Regression Cross-Validation Score for Metro Dataset: {cv_mean_ridge_metro:.4f}")
```

```
Ridge Regression R² Score for Metro Dataset: 0.0234
Ridge Regression Cross-Validation Score for Metro Dataset: -2.5344
```

Lasso-Model on Metro data

```python
# Feature Scaling
scaler = StandardScaler()
X_metro_scaled = scaler.fit_transform(X_metro)

lasso_metro = Lasso(alpha=1.0, max_iter=10000)

# Train the model on the training data
lasso_metro.fit(X_train_metro, y_train_metro)

# Make predictions on the test data
y_pred_lasso_metro = lasso_metro.predict(X_test_metro)

# Evaluate the model using R² score
r2_lasso_metro = r2_score(y_test_metro, y_pred_lasso_metro)
print(f"Lasso Regression R² Score for Metro Dataset: {r2_lasso_metro:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_lasso_metro = cross_val_score(lasso_metro, X_metro_scaled, y_metro, cv=5)
cv_mean_lasso_metro = np.mean(cv_scores_lasso_metro)
print(f"Lasso Regression Cross-Validation Score for Metro Dataset: {cv_mean_lasso_metro:.4f}")
```

```
Lasso Regression R² Score for Metro Dataset: 0.0234
Lasso Regression Cross-Validation Score for Metro Dataset: -0.3194
```

LinearRegression-Model on Auto data

```python
# Drop rows with missing target values
df_autos_clean = df_autos_clean.dropna(subset=['price'])

# Convert categorical variables to dummy variables
df_autos_clean = pd.get_dummies(df_autos_clean, columns=['make', 'fuel-type', 'aspiration', 'num-of-doors',
                                                         'body-style', 'drive-wheels', 'engine-location',
                                                         'engine-type', 'num-of-cylinders', 'fuel-system'], drop_first=True)

# Prepare the data for regression
X_autos = df_autos_clean.drop(columns=['price'])  # Features
y_autos = df_autos_clean['price']  # Target variable

# Split the data into training and testing sets
X_train_autos, X_test_autos, y_train_autos, y_test_autos = train_test_split(X_autos, y_autos, test_size=0.2, random_state=42)

lr_autos = LinearRegression()

# Train the model on the training data
lr_autos.fit(X_train_autos, y_train_autos)

# Make predictions on the test data
y_pred_autos = lr_autos.predict(X_test_autos)

# Evaluate the model using R² score
r2_autos = r2_score(y_test_autos, y_pred_autos)
print(f"Linear Regression R² Score for Autos Dataset: {r2_autos:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_autos = cross_val_score(lr_autos, X_autos, y_autos, cv=5)
cv_mean_autos = np.mean(cv_scores_autos)
print(f"Linear Regression Cross-Validation Score for Autos Dataset: {cv_mean_autos:.4f}")
```

```
Linear Regression R² Score for Autos Dataset: 0.8902
Linear Regression Cross-Validation Score for Autos Dataset: -0.1857
```

Ridge-Model on Auto data

```python
ridge_autos = Ridge(alpha=1.0)

# Train the model on the training data
ridge_autos.fit(X_train_autos, y_train_autos)

# Make predictions on the test data
y_pred_ridge_autos = ridge_autos.predict(X_test_autos)

# Evaluate the model using R² score
r2_ridge_autos = r2_score(y_test_autos, y_pred_ridge_autos)
print(f"Ridge Regression R² Score for Autos Dataset: {r2_ridge_autos:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_ridge_autos = cross_val_score(ridge_autos, X_autos, y_autos, cv=5)
cv_mean_ridge_autos = np.mean(cv_scores_ridge_autos)
print(f"Ridge Regression Cross-Validation Score for Autos Dataset: {cv_mean_ridge_autos:.4f}")
```

```
Ridge Regression R² Score for Autos Dataset: 0.8806
Ridge Regression Cross-Validation Score for Autos Dataset: 0.3715
```

Lasso-Model on Auto data

```python
# Feature Scaling
scaler = StandardScaler()
X_autos_scaled = scaler.fit_transform(X_autos)

lasso_autos = Lasso(alpha=1.0, max_iter=10000)

# Train the model on the training data
lasso_autos.fit(X_train_autos, y_train_autos)

# Make predictions on the test data
y_pred_lasso_autos = lasso_autos.predict(X_test_autos)

# Evaluate the model using R² score
r2_lasso_autos = r2_score(y_test_autos, y_pred_lasso_autos)
print(f"Lasso Regression R² Score for Autos Dataset: {r2_lasso_autos:.4f}")

# Perform cross-validation and calculate the mean cross-validation score
cv_scores_lasso_autos = cross_val_score(lasso_autos, X_autos_scaled, y_autos, cv=5)
cv_mean_lasso_autos = np.mean(cv_scores_lasso_autos)
print(f"Lasso Regression Cross-Validation Score for Autos Dataset: {cv_mean_lasso_autos:.4f}")
```

```
Lasso Regression R² Score for Autos Dataset: 0.8894
Lasso Regression Cross-Validation Score for Autos Dataset: 0.1326
```

Comparison of each model for Bike data

```python
metrics_bike = {
    'Model': ['Linear Regression', 'Ridge Regression', 'Lasso Regression'],
    'R² Score': [r2_bike, r2_ridge_bike, r2_lasso_bike],
    'Cross-Validation Score': [cv_mean_bike, cv_mean_ridge_bike, cv_mean_lasso_bike]
}

df_metrics_bike = pd.DataFrame(metrics_bike)
print("Bike Data Model Comparison :\n")
print(df_metrics_bike)
```

```
Bike Data Model Comparison :

               Model  R² Score  Cross-Validation Score
0  Linear Regression  0.499472               -1.964897
1   Ridge Regression  0.486895               -1.871240
2   Lasso Regression  0.499442               -1.964756
```

Comparison of each model for Metro data

```python
metrics_metro = {
    'Model': ['Linear Regression', 'Ridge Regression', 'Lasso Regression'],
    'R² Score': [r2_metro, r2_ridge_metro, r2_lasso_metro],
    'Cross-Validation Score': [cv_mean_metro, cv_mean_ridge_metro, cv_mean_lasso_metro]
}

df_metrics_metro = pd.DataFrame(metrics_metro)
print("Metro Data Model Comparison:")
print(df_metrics_metro)
```

```
Metro Data Model Comparison:
               Model  R² Score  Cross-Validation Score
0  Linear Regression  0.023424               -2.534509
1   Ridge Regression  0.023425               -2.534377
2   Lasso Regression  0.023427               -0.319376
```

Comparison of each model for Auto data

```python
metrics_autos = {
    'Model': ['Linear Regression', 'Ridge Regression', 'Lasso Regression'],
    'R² Score': [r2_autos, r2_ridge_autos, r2_lasso_autos],
    'Cross-Validation Score': [cv_mean_autos, cv_mean_ridge_autos, cv_mean_lasso_autos]
}

df_metrics_autos = pd.DataFrame(metrics_autos)
print("Autos Data Model Comparison:")
print(df_metrics_autos)
```

```
Autos Data Model Comparison:
               Model  R² Score  Cross-Validation Score
0  Linear Regression  0.890177               -0.185669
1   Ridge Regression  0.880586                0.371468
2   Lasso Regression  0.889406                0.132572
```

```python
# Combine metrics from all datasets
overall_comparison = pd.concat([
    df_metrics_bike.assign(Dataset='Bike Data'),
    df_metrics_metro.assign(Dataset='Metro Data'),
    df_metrics_autos.assign(Dataset='Autos Data')
])

print("Overall Model Comparison:")
print(overall_comparison)
```

```
Overall Model Comparison:
               Model  R² Score  Cross-Validation Score     Dataset
0  Linear Regression  0.499472               -1.964897   Bike Data
1   Ridge Regression  0.486895               -1.871240   Bike Data
2   Lasso Regression  0.499442               -1.964756   Bike Data
0  Linear Regression  0.023424               -2.534509  Metro Data
1   Ridge Regression  0.023425               -2.534377  Metro Data
2   Lasso Regression  0.023427               -0.319376  Metro Data
0  Linear Regression  0.890177               -0.185669  Autos Data
1   Ridge Regression  0.880586                0.371468  Autos Data
2   Lasso Regression  0.889406                0.132572  Autos Data
```