```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.metrics import r2_score, mean_squared_error
import zipfile
import io
import requests

# Load the dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip'
response = requests.get(url)
zip_file = zipfile.ZipFile(io.BytesIO(response.content))


file_name = 'day.csv'

# Directly read the chosen CSV file into a DataFrame
data = pd.read_csv(zip_file.open(file_name))

# Preview the first few rows of the dataset
data.head()
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	regis
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	

Next steps: Generate code with data View recommended plots New interactive sheet

```
data.dropna(inplace=True)

# Check for duplicates and remove them
data.drop_duplicates(inplace=True)
```

```
# Feature selection: Let's choose some relevant features
X = data[['temp']]  # Simple linear regression on 'temp'
X_multi = data[['temp', 'atemp', 'hum', 'windspeed']]  # Multiple Linear Regression
y = data['cnt']

# Feature scaling (optional but recommended for some algorithms)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_multi_scaled = scaler.fit_transform(X_multi)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
X_multi_train, X_multi_test, y_multi_train, y_multi_test = train_test_split(X_multi_scaled, y, test_size=0.2, random_state=42)

# 1. Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)

# 2. Multiple Linear Regression
multi_lin_reg = LinearRegression()
multi_lin_reg.fit(X_multi_train, y_multi_train)
y_pred_multi_lin = multi_lin_reg.predict(X_multi_test)

# 3. Polynomial Regression (degree 2)
poly_features = PolynomialFeatures(degree=2)
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)
poly_reg = LinearRegression()
poly_reg.fit(X_poly_train, y_train)
y_pred_poly = poly_reg.predict(X_poly_test)

# 4. Multiple Polynomial Regression (degree 2)
X_poly_multi_train = poly_features.fit_transform(X_multi_train)
X_poly_multi_test = poly_features.transform(X_multi_test)
poly_multi_reg = LinearRegression()
poly_multi_reg.fit(X_poly_multi_train, y_multi_train)
```

```
y_pred_poly_multi = poly_multi_reg.predict(X_poly_multi_test)

# Evaluate the models
r2_lin = r2_score(y_test, y_pred_lin)
mse_lin = mean_squared_error(y_test, y_pred_lin)

r2_multi_lin = r2_score(y_multi_test, y_pred_multi_lin)
mse_multi_lin = mean_squared_error(y_multi_test, y_pred_multi_lin)

r2_poly = r2_score(y_test, y_pred_poly)
mse_poly = mean_squared_error(y_test, y_pred_poly)

r2_poly_multi = r2_score(y_multi_test, y_pred_poly_multi)
mse_poly_multi = mean_squared_error(y_multi_test, y_pred_poly_multi)

# Print results
print("Linear Regression: R2 =", r2_lin, "MSE =", mse_lin)
print("Multiple Linear Regression: R2 =", r2_multi_lin, "MSE =", mse_multi_lin)
print("Polynomial Regression (degree 2): R2 =", r2_poly, "MSE =", mse_poly)
print("Multiple Polynomial Regression (degree 2): R2 =", r2_poly_multi, "MSE =", mse_poly_multi)
```

```
Linear Regression: R2 = 0.40371020554910975 MSE = 2391051.8856316973
Multiple Linear Regression: R2 = 0.4994717184081341 MSE = 2007059.4912903379
Polynomial Regression (degree 2): R2 = 0.393648911197503 MSE = 2431396.4916524296
Multiple Polynomial Regression (degree 2): R2 = 0.573097817195138 MSE = 1711827.501786835
```

Multiple Polynomial Regression has high R2 Score and low MSE hence it is good model.

Metro Interstate Traffic Volume Data Set

```
!pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.7-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.8.30)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucim
Downloading ucimlrepo-0.0.7-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.7
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
metro_interstate_traffic_volume = fetch_ucirepo(id=492)

# data (as pandas dataframes)
X = metro_interstate_traffic_volume.data.features
y = metro_interstate_traffic_volume.data.targets
```

```
import pandas as pd

# Combine X and y into a single DataFrame to handle missing values and duplicates
df = X.copy()
df['traffic_volume'] = y

# Drop rows with missing values
df = df.dropna()



# Check for duplicates
df = df.drop_duplicates()

df = df.drop(columns=['date_time'])
df = pd.get_dummies(df, columns=['weather_main', 'weather_description'], drop_first=True)

df.head()
```

	holiday	temp	rain_1h	snow_1h	clouds_all	traffic_volume	weather_main_Clouds	weather_main_Haze	weather_main_Mist	w
126	Columbus Day	273.08	0.0	0.0	20	455	True	False	False	
1123	Veterans Day	288.12	0.0	0.0	87	1000	False	False	False	
1370	Thanksgiving Day	278.54	0.0	0.0	20	919	False	False	True	
2360	Christmas Day	264.40	0.0	0.0	90	803	True	False	False	
2559	New Years Day	263.49	0.0	0.0	58	1439	True	False	False	

5 rows × 23 columns

```
# Define features (X) and target variable (y)
X = df[['temp']]  # Simple linear regression on 'temp'
X_multi = df[['temp', 'rain_1h', 'snow_1h', 'clouds_all']]  # Multiple Linear Regression
y = df['traffic_volume']

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_multi_scaled = scaler.fit_transform(X_multi)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
X_multi_train, X_multi_test, y_multi_train, y_multi_test = train_test_split(X_multi_scaled, y, test_size=0.2, random_state=42)

# 1. Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)

# 2. Multiple Linear Regression
multi_lin_reg = LinearRegression()
multi_lin_reg.fit(X_multi_train, y_multi_train)
y_pred_multi_lin = multi_lin_reg.predict(X_multi_test)

# 3. Polynomial Regression (degree 2)
poly_features = PolynomialFeatures(degree=2)
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)
poly_reg = LinearRegression()
poly_reg.fit(X_poly_train, y_train)
y_pred_poly = poly_reg.predict(X_poly_test)

# 4. Multiple Polynomial Regression (degree 2)
X_poly_multi_train = poly_features.fit_transform(X_multi_train)
X_poly_multi_test = poly_features.transform(X_multi_test)
poly_multi_reg = LinearRegression()
poly_multi_reg.fit(X_poly_multi_train, y_multi_train)
y_pred_poly_multi = poly_multi_reg.predict(X_poly_multi_test)

# Evaluate the models
r2_lin = r2_score(y_test, y_pred_lin)
mse_lin = mean_squared_error(y_test, y_pred_lin)

r2_multi_lin = r2_score(y_multi_test, y_pred_multi_lin)
mse_multi_lin = mean_squared_error(y_multi_test, y_pred_multi_lin)

r2_poly = r2_score(y_test, y_pred_poly)
mse_poly = mean_squared_error(y_test, y_pred_poly)

r2_poly_multi = r2_score(y_multi_test, y_pred_poly_multi)
mse_poly_multi = mean_squared_error(y_multi_test, y_pred_poly_multi)

# Print results
print("Linear Regression: R2 =", r2_lin, "MSE =", mse_lin)
print("Multiple Linear Regression: R2 =", r2_multi_lin, "MSE =", mse_multi_lin)
print("Polynomial Regression (degree 2): R2 =", r2_poly, "MSE =", mse_poly)
print("Multiple Polynomial Regression (degree 2): R2 =", r2_poly_multi, "MSE =", mse_poly_multi)
```

```
Linear Regression: R2 = -0.37889749644174864 MSE = 87603.98519742863
Multiple Linear Regression: R2 = -0.4357725527998362 MSE = 91217.36589334992
```

```
Polynomial Regression (degree 2): R2 = -0.2350827307815635 MSE = 78467.15912110488
Multiple Polynomial Regression (degree 2): R2 = -0.338745693877079 MSE = 85053.06467824413
```

Polynomial Regression is performing good.