

**AIM:**

To clean, preprocess, and visualize Oil Price data, focusing on trend analysis and handling missing values.

**ALGORITHM:**

1. Load the oil price data from the CSV file.
2. Parse the date column and set it as the index.
3. Handle missing values by filling them with forward fill.
4. Convert columns like Open, Close, Volume to numeric values.
5. Compute moving averages (7-day and 30-day) for trend analysis.
6. Drop any rows with NaN values created during moving average computation.
7. Visualize the closing price along with the moving averages using a line plot.

**CODE:**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import numpy as np

# 1. Load the dataset

df = pd.read\_csv(r'C:\Users\Lenovo\Downloads\crude-oil-price.csv') # Update the path

# 2. Convert the 'date' column to datetime format

df['date'] = pd.to\_datetime(df['date'], errors='coerce') # Convert 'date' to datetime format

# Remove any rows where the 'date' or 'price' columns are missing

df\_cleaned = df.dropna(subset=['date', 'price'])

# 3. Plot the oil prices as a histogram (Bar Plot)

plt.figure(figsize=(8,6))

sns.histplot(df\_cleaned['price'], bins=20, kde=False, color='blue') # Plot the histogram

plt.title('Distribution of Crude Oil Prices') # Title of the plot

plt.xlabel('Price ($)') # Label for the x-axis

plt.ylabel('Frequency') # Label for the y-axis

plt.show()

# 4. Create a log-transformed feature of the price

df\_cleaned['log\_price'] = np.log(df\_cleaned['price']) # Create log-transformed price feature

# 5. Split the data into training and testing sets (we'll predict 'price' using 'date' for simplicity)

df\_cleaned['date\_ordinal'] = df\_cleaned['date'].apply(lambda x: x.toordinal()) # Convert date to ordinal (numeric format)

X = df\_cleaned[['date\_ordinal']] # Features (date in numeric form)

y = df\_cleaned['price'] # Target variable (price)

# Splitting the dataset into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 6. Train a Linear Regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# 7. Predict on the test set

y\_pred = model.predict(X\_test)

# 8. Plot Actual vs. Predicted prices

plt.figure(figsize=(8,6))

plt.scatter(y\_test, y\_pred, color='green') # Scatter plot for actual vs predicted

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2) # Diagonal line (y=x)

plt.title('Actual vs Predicted Crude Oil Prices')

plt.xlabel('Actual Price ($)')

plt.ylabel('Predicted Price ($)')

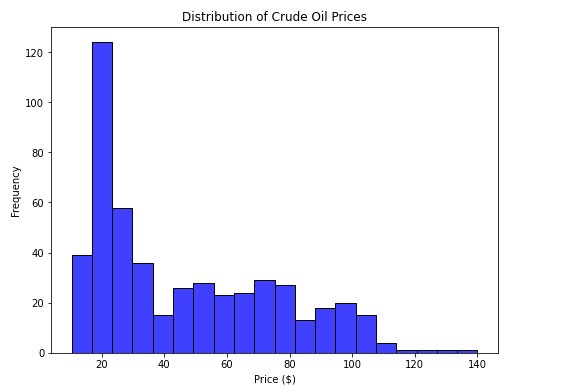
plt.show()

# 9. Calculate RMSE (Root Mean Squared Error)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f'RMSE: {rmse:.

**OUTPUT:**

****

**RESULT:**

Thus the program has been completed and verified successfully.