Task - 1

In [2]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor

# Load the datasets
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

# Convert date column to datetime format
train['date'] = pd.to_datetime(train['date'])
test['date'] = pd.to_datetime(test['date'])

# Extract date features
train['year'] = train['date'].dt.year
train['month'] = train['date'].dt.month
train['day'] = train['date'].dt.day
train['day_of_week'] = train['date'].dt.dayofweek

test['year'] = test['date'].dt.year
test['month'] = test['date'].dt.month
test['day'] = test['date'].dt.day
test['day_of_week'] = test['date'].dt.dayofweek

# Create lag features
for lag in range(1, 8):
    train[f'lag_{lag}'] = train.groupby('Item Id')['units'].shift(lag)

# Drop rows with NaN values created by lag features before applying rolli
train.dropna(subset=[f'lag_{lag}' for lag in range(1, 8)], inplace=True)

# Create rolling window features
train['rolling_mean_7'] = train.groupby('Item Id')['units'].transform(lam
train['rolling_std_7'] = train.groupby('Item Id')['units'].transform(lamb

# Drop rows with NaN values created by rolling window features
train.dropna(subset=['rolling_mean_7', 'rolling_std_7'], inplace=True)

# Handle any remaining NaN values by filling them with the mean of their
train.fillna(train.mean(), inplace=True)

# Check for infinite or excessively large values and replace them
train.replace([np.inf, -np.inf], np.nan, inplace=True)
train.fillna(train.mean(), inplace=True)

# Define the feature columns and target column
feature_cols = ['ad_spend', 'unit_price', 'year', 'month', 'day', 'day_of
               [f'lag_{lag}' for lag in range(1, 8)] + \
               ['rolling_mean_7', 'rolling_std_7']
target_col = 'units'

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(train[feature_cols], tr
# Initialize and train the model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the validation set
val_preds = model.predict(X_val)

# Evaluate the model
mse = mean_squared_error(y_val, val_preds)
print(f'Validation MSE: {mse}')

# Prepare the test set features
# Since 'units' column is not available in the test set, we'll use mean v
for lag in range(1, 8):
    test[f'lag_{lag}'] = train.groupby('Item Id')['units'].transform(lamb

# Ensure lag features align correctly in the test set
test.fillna(test.mean(), inplace=True)

# For rolling mean and std, we can't calculate directly on test set witho
# We'll use mean values from the training set for these features as well
test['rolling_mean_7'] = train.groupby('Item Id')['units'].transform(lamb
test['rolling_std_7'] = train.groupby('Item Id')['units'].transform(lambd

# Ensure rolling window features align correctly in the test set
test.fillna(test.mean(), inplace=True)

# Check for infinite or excessively large values and replace them
test.replace([np.inf, -np.inf], np.nan, inplace=True)
test.fillna(test.mean(), inplace=True)

# Make predictions on the test set
test_preds = model.predict(test[feature_cols])

# Create a submission DataFrame
submission = test[['date', 'Item Id']].copy()
submission['predicted_units'] = test_preds

# Save the submission file
submission.to_csv('C:/Users/saicharan/OneDrive/Desktop/Nap Queens/Task1/s
print("Submission file created.")
```

```
Validation MSE: 1186.77385064152
Submission file created.
```

Task-2

In [3]:

```
# Convert date column to datetime format
train['date'] = pd.to_datetime(train['date'])
test['date'] = pd.to_datetime(test['date'])

# Extract date features
train['year'] = train['date'].dt.year
train['month'] = train['date'].dt.month
train['day'] = train['date'].dt.day
train['day_of_week'] = train['date'].dt.dayofweek

test['year'] = test['date'].dt.year
```

```
test['month'] = test['date'].dt.month
test['day'] = test['date'].dt.day
test['day_of_week'] = test['date'].dt.dayofweek

# Create lag features
for lag in range(1, 8):
    train[f'lag_{lag}'] = train.groupby('Item Id')['units'].shift(lag)

# Drop rows with NaN values created by lag features before applying rolli
train.dropna(subset=[f'lag_{lag}' for lag in range(1, 8)], inplace=True)

# Create rolling window features
train['rolling_mean_7'] = train.groupby('Item Id')['units'].transform(lam
train['rolling_std_7'] = train.groupby('Item Id')['units'].transform(lamb

# Drop rows with NaN values created by rolling window features
train.dropna(subset=['rolling_mean_7', 'rolling_std_7'], inplace=True)

# Handle any remaining NaN values by filling them with the mean of their
train.fillna(train.mean(), inplace=True)

# Check for infinite or excessively large values and replace them
train.replace([np.inf, -np.inf], np.nan, inplace=True)
train.fillna(train.mean(), inplace=True)

# Define the feature columns and target column, excluding 'ad_spend'
feature_cols = ['unit_price', 'year', 'month', 'day', 'day_of_week'] + \
               [f'lag_{lag}' for lag in range(1, 8)] + \
               ['rolling_mean_7', 'rolling_std_7']
target_col = 'units'

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(train[feature_cols], tr

# Initialize and train the model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the validation set
val_preds = model.predict(X_val)

# Evaluate the model
mse = mean_squared_error(y_val, val_preds)
print(f'Validation MSE: {mse}')

# Prepare the test set features
# Since 'units' column is not available in the test set, we'll use mean v
for lag in range(1, 8):
    test[f'lag_{lag}'] = train.groupby('Item Id')['units'].transform(lamb

# Ensure lag features align correctly in the test set
test.fillna(test.mean(), inplace=True)

# For rolling mean and std, we can't calculate directly on test set witho
# We'll use mean values from the training set for these features as well
test['rolling_mean_7'] = train.groupby('Item Id')['units'].transform(lamb
test['rolling_std_7'] = train.groupby('Item Id')['units'].transform(lambd

# Ensure rolling window features align correctly in the test set
test.fillna(test.mean(), inplace=True)
```

```python
# Check for infinite or excessively large values and replace them
test.replace([np.inf, -np.inf], np.nan, inplace=True)
test.fillna(test.mean(), inplace=True)

# Make predictions on the test set
test_preds = model.predict(test[feature_cols])

# Create a submission DataFrame
submission = test[['date', 'Item Id']].copy()
submission['predicted_units'] = test_preds

# Save the submission file
submission.to_csv('C:/Users/saicharan/OneDrive/Desktop/Nap Queens/Task2/s
print("Submission file created.")
```

```
Validation MSE: 3061.47883598121
Submission file created.
```