AKULA MALLIKHARJUNA
akulamallikharjuna2001@gmail.com

Predicted results:

Task1:

```
import numpy as np
# Prepare data for modeling
features = ['year', 'month', 'day', 'dayofweek', 'ad_spend', 'unit_price', 'units_lag1', 'units_lag7', 'units_lag30'
X = train_data[features]
y = train_data['units']

# Check for NaN, infinity, or very large values in y
print(f"Before cleaning: NaNs: {y.isna().sum()}, Infinities: {np.isinf(y).sum()}, Too large values: {(y > 1e6).sum()

# Handle NaN, infinity, or very large values
y = y.replace([np.inf, -np.inf], np.nan)
y = y.fillna(y.median())
y = y.clip(upper=y.quantile(0.99))

print(f"After cleaning: NaNs: {y.isna().sum()}, Infinities: {np.isinf(y).sum()}, Too large values: {(y > 1e6).sum()}

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Train an XGBoost model
model = XGBRegressor(objective='reg:squarederror')
model.fit(X_train, y_train)

# Validate the model
y_pred = model.predict(X_val)
mse = mean_squared_error(y_val, y_pred)
print(f'Validation Mean Square Error: {mse}')
```

```
Before cleaning: NaNs: 17898, Infinities: 0, Too large values: 0
After cleaning: NaNs: 0, Infinities: 0, Too large values: 0
Validation Mean Square Error: 40.44769716125222
```

Task 2:

```
# Prepare data for modeling without 'ad_spend'
features = ['year', 'month', 'day', 'dayofweek', 'unit_price', 'units_lag1', 'units_lag7', 'units_lag30', 'units_rol
X = train_df[features]
y = train_df['units']

# Check for NaN, infinity, or very large values in y
print(f"Before cleaning: NaNs: {y.isna().sum()}, Infinities: {np.isinf(y).sum()}, Too large values: {(y > 1e6).sum()

# Handle NaN, infinity, or very large values
y = y.replace([np.inf, -np.inf], np.nan)
y = y.fillna(y.median())
y = y.clip(upper=y.quantile(0.99))

print(f"After cleaning: NaNs: {y.isna().sum()}, Infinities: {np.isinf(y).sum()}, Too large values: {(y > 1e6).sum()}

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Train an XGBoost model
model = XGBRegressor(objective='reg:squarederror')
model.fit(X_train, y_train)

# Validate the model
y_pred = model.predict(X_val)
mse = mean_squared_error(y_val, y_pred)
print(f'Validation Mean Square Error: {mse}')
```

```
Before cleaning: NaNs: 0, Infinities: 0, Too large values: 0
After cleaning: NaNs: 0, Infinities: 0, Too large values: 0
Validation Mean Square Error: 283.22200933221495
```