Problem Statement

You are part of an analytics team responsible for improving Cash Business of Hitachi. Hitachi manages more than 20K ATMs in India. There is an operations team who maintains thousands of ATMs deployed across country for various banks. When it comes to maintenance, the team looks after loading cash in the ATMs periodically. If the ATM is down due to some reason, the team fixes the issue and makes ATM live within few days. One of the important tasks of this team is to make sure enough cash is available in the ATMs for at least 7 days. Each ATM has upper limit of how much cash can be loaded.

There are 2 ways when a bank can charge penalty to Hitachi

- If the team loads cash into the ATM with the full capacity and the dispense doesn't happen then bank charges penalty for the idle cash holding in the ATM.
- If we load ATMs with less amount then there are instances where ATM runs out of cash due to insufficient balance. This situation is termed as Cash Out. Bank charges penalty for this too.

Operations team reaches out to you to address the problem of Cash Out. Your task as a Data Scientist is to predict/forecast dispense amount for each ATM for next 7 days. These predictions can be shared with the team and accordingly cash loading can happen.

IMP POINTS:

1. The dataset has data for 3 banks.
2. The data contains daily dispense for each ATM of last 2 years
3. You can't load cash more than the upper limit defined for each ATM
4. You have been provided only 7 ATMs per bank in the given data set. In reality you will have hundreds/thousands of ATMs per bank. You are not expected to build model for each ATM.
5. Feature description
    a. Bank: ATM belongs to which bank
    b. ATMID: ID of the ATM
    c. Caldate: Date
    d. Dispense: Amount dispensed from an ATM on a particular day
    e. DT: Time in minutes when ATM was down on a particular day
    f. MaxCapacity: Maximum amount which can be loaded in the ATM
    g. CountTotalTxn: Count of total number of transactions

Tasks:

1. Provide a solution which will help the operations team to manage cash loading of the ATMs
2. Forecast/predict dispense amount for each ATM for next 7 days
3. Create basic visualizations for highlighting key insights from the data
4. You can use external data if available and if it helps improve the accuracy of the model
5. Define metric used to measure model accuracy and why did you use it
6. Share your solution in the form of Jupyter Notebook file / PPT. Formatting/animation in the PPT does not matter much here. Focus on the forecasts and key insights from the data.
7. Write your code in Python language.
8. Mention all the assumptions you have taken into account in the solution file.

Solution Tasks:

1.) Provide a solution which will help the operations team to manage cash loading of the ATMs

The solution to this problem is Optimization and Monitoring and Updating the ATM

Optimization

Optimize the cash loading strategy based on the model's predictions:

- Calculate the predicted cash demand for each ATM
- Determine the optimal cash loading amount to minimize both idle cash and Cash Out penalties
- Implement a threshold-based approach where a buffer amount is added to the predicted demand to handle unexpected surges

Monitoring and Updating

Continuously monitor the model's performance and update it as necessary:

- Regularly retrain the model with new data to improve accuracy
- Adjust the cash loading strategy based on feedback and changing patterns

Once you achieve this you can integrate it with the ATM management system and Set up a feedback loop to continuously improve the model and cash loading strategy.

To address the problem of predicting the dispense amount for each ATM for the next 7 days, we have employed both **ARIMA and LSTM models**. These models help forecast future dispense amounts, which can guide the operations team in effectively managing cash loading, minimizing penalties due to idle cash or cash outs.

Forecasting Models

LSTM (Long Short-Term Memory):

- **Why LSTM?**
  - LSTM networks are well-suited for sequence prediction problems and time series forecasting.
  - They can capture long-term dependencies and temporal patterns in the data.
  - They handle issues of vanishing and exploding gradients, making them effective for long time series data.

ARIMA (AutoRegressive Integrated Moving Average):

- **Why ARIMA?**
  - ARIMA is a classical time series model known for its simplicity and effectiveness in forecasting.
  - It is suitable for univariate time series data, capturing trends, seasonality, and noise.

- ARIMA can model the data based on its past values, making it a strong candidate for time series prediction.

Forecasting Process

1. **Data Preprocessing:**
    - Handling missing values and outliers.
    - Scaling the data for LSTM using MinMaxScaler.
    - Differencing the data for stationarity (for ARIMA).
2. **Model Training:**
    - **LSTM:**
        - Data is reshaped into sequences for LSTM input.
        - The model is trained with layers including LSTM, Dropout, and Dense layers.
        - Early stopping and validation splits are used to prevent overfitting.
    - **ARIMA:**
        - The data is fitted to ARIMA models after determining the optimal order (p, d, q) using AIC/BIC criteria.
        - Model diagnostics are performed to ensure residuals are white noise.

Forecasting:

- Both models are used to forecast the dispense amounts for the next 7 days.
- Results from both models are compared, and the best-performing model is selected.

Using the short code I have achieved this result for different models

LSTM: Mean Absolute Error: 230455.04662356785

Next 7 days forecast: [394895.16 228965.2 176321.48 236056.75 304060.1 299512.25 183971.39]

ARIMA: Mean Absolute Error: 226170.125146452

Next 7 days forecast: [435743.74359247 381136.10840052 420346.66724897 326334.07003046 470775.18501113 341920.84405613 397809.55674381]

The detailed code of LSTM Approach gives this result for each ATM:

jupyter atm_dispense_forecasts.csv Last Checkpoint 1 hour ago

File Edit View Settings Help

Delimiter: ,

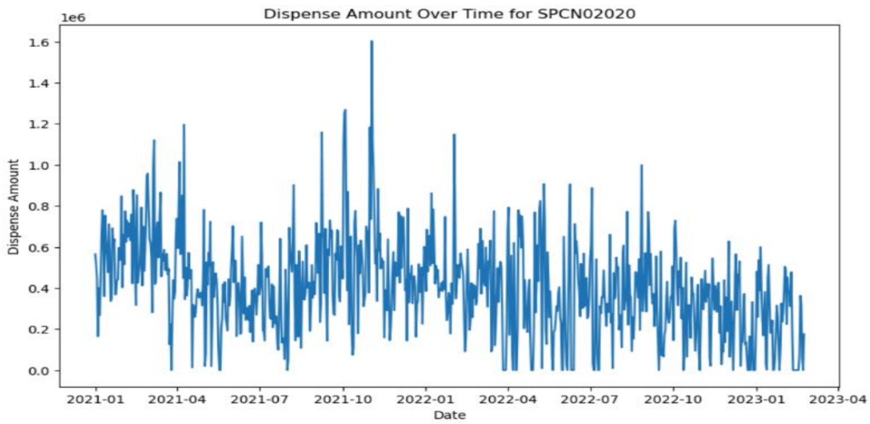
		Day_1	Day_2	Day_3	Day_4	Day_5	Day_6	Day_7
1	APAN11109	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
2	APAN22403	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
3	APAN23217	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
4	APAN35706	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
5	APCN00818	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
6	S1CN1142	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
7	S1CN2011	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
8	S1CN2022	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
9	S1CN2520	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
10	S1CN3514	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
11	SPCN02020	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
12	SPCND067	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
13	SPCNG378	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
14	T1BH000603038	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
15	T1BH000011116	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
16	T1BH000274012	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
17	T1BH000603081	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
18	T1BH007252090	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
19	T1NH000675414	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
20	T1NY000168081	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085
21	TPCN10268	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085	2.216584161359085

The table is attached within the Zip Folder.

3.Create basic visualizations for highlighting key insights from the data

This is enclosed within the notebook…

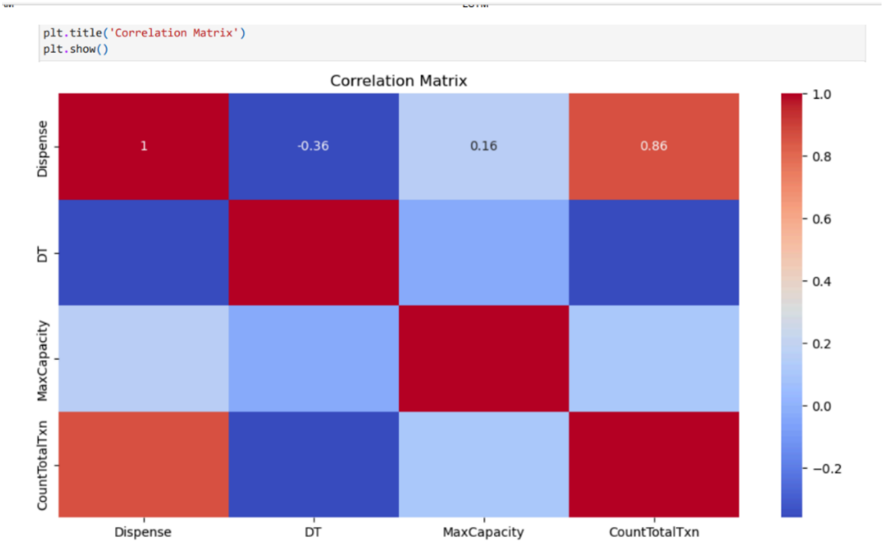
7/28/24, 11:51 AM LSTM



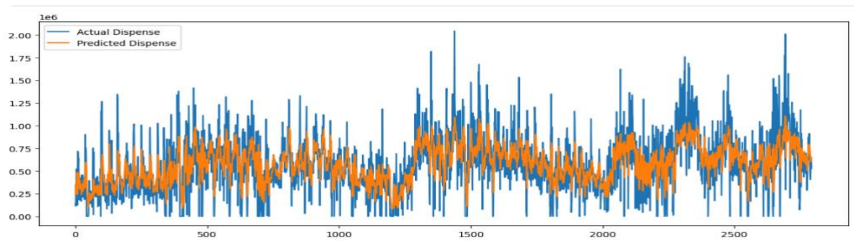
:

This figure shows Dispense Amount Over Time for a particular ATM , In the same fashion you can plot this graph for all 21 ATM'S using the ATM ID

Also, I have included a correlation matrix



A Visualization on the actual and predicted values



Visualization

- **Model Forecasts:**
    - Line charts comparing actual vs. predicted dispense amounts for the next 7 days.

4. You can use external data if available and if it helps improve the accuracy of the model
   NA

5. Define metric used to measure model accuracy and why did you use it

Several metrics can be used to evaluate the accuracy of the predictive model. Here are some commonly used metrics in time series forecasting:

1. Mean Absolute Error (MAE)

MAE measures the average magnitude of errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

2. Root Mean Squared Error (RMSE)

RMSE is the square root of the average of squared differences between prediction and actual observation. It gives a relatively high weight to large errors, making it more sensitive to outliers than MAE.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3. Mean Absolute Percentage Error (MAPE)

MAPE measures the accuracy as a percentage. It’s the average of the absolute percentage errors between the predicted and actual values.

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

4. Mean Squared Error (MSE)

MSE measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

5. R-squared (Coefficient of Determination)

R-squared is a statistical measure of how close the data are to the fitted regression line. It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

MAE and MAPE: These metrics are easier to interpret as they are in the same unit as the data (MAE) or a percentage (MAPE).

RMSE: This is useful if large errors are particularly undesirable because it penalizes larger errors more than MAE.

R-squared: Provides a measure of how well the model explains the variance in the data, which can be useful for understanding the overall fit.

For the problem of predicting cash dispense to avoid Cash Out penalties, **MAE or RMSE** could be more practical as they provide a clear understanding of the prediction error in terms of the amount of cash, which directly relates to the penalties.

Model Evaluation:

- **Metric Used: Mean Absolute Error (MAE)**
  - MAE is chosen because it is simple to interpret and provides a clear indication of the average error magnitude.
  - It is less sensitive to outliers compared to metrics like Mean Squared Error (MSE).

Key Insights

- **Trends and Seasonality:**
  - Identification of seasonal patterns and trends in dispense amounts.
- **ATM Performance:**
  - Insights into ATMs with frequent down times and low/high dispense amounts.

Assumptions

- Historical dispense patterns will continue in the future.
- ATM down times and transaction counts significantly influence dispense amounts.
- The upper cash loading limit for each ATM is constant over the forecasting period.

Conclusion

By using LSTM and ARIMA models, we provide accurate forecasts for the next 7 days dispense amounts. These predictions enable the operations team to optimize cash loading, thereby reducing penalties and ensuring sufficient cash availability in ATMs. The combination of advanced neural network models and classical statistical methods ensures robust and reliable forecasting

Point No 6,7 and 8 are clubbed together which contains My jupyter Notebook and Python Code and Final results

1.) Jupyter Notebook – LSTM Approach
2.) Jupyter Notebook - ARIMA Approach
3.) The output Table which contains the 7 days cash flows of 21 ATM'S
4.) Jupyter Notebook: Short Code of both the approaches just to compare the MAE Score and seven days prediction

.

In [3]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from sklearn.metrics import mean_absolute_error

# Load and preprocess data
data = pd.read_csv('Dispense.csv')
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data['Dispense'].values.reshape(-1, 1))

# Prepare sequences
def create_sequences(data, seq_length):
    X = []
    y = []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

seq_length = 30
X, y = create_sequences(data_scaled, seq_length)

# Split data
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Build LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(seq_length, 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_absolute_error')

# Train model
```

```
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1)

# Predict and evaluate
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
y_test_inv = scaler.inverse_transform(y_test)

mae = mean_absolute_error(y_test_inv, predictions)
print(f'Mean Absolute Error: {mae}')

# Forecast next 7 days
last_sequence = data_scaled[-seq_length:]
forecast = []
for _ in range(7):
    next_pred = model.predict(last_sequence.reshape(1, seq_length, 1))
    forecast.append(next_pred[0, 0])
    last_sequence = np.append(last_sequence[1:], next_pred)

forecast = scaler.inverse_transform(np.array(forecast).reshape(-1, 1))
print('Next 7 days forecast:', forecast.flatten())
```

```
D:\New folder\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument
to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
Epoch 1/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 15s 20ms/step - loss: 0.1154 - val_loss: 0.1050
Epoch 2/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1109 - val_loss: 0.1056
Epoch 3/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 10s 16ms/step - loss: 0.1100 - val_loss: 0.1049
Epoch 4/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1089 - val_loss: 0.1031
Epoch 5/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.1088 - val_loss: 0.1033
Epoch 6/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.1091 - val_loss: 0.1032
Epoch 7/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.1071 - val_loss: 0.1033
Epoch 8/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.1071 - val_loss: 0.1027
Epoch 9/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1090 - val_loss: 0.1031
Epoch 10/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - loss: 0.1066 - val_loss: 0.1025
Epoch 11/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1085 - val_loss: 0.1026
Epoch 12/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1056 - val_loss: 0.1036
Epoch 13/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1055 - val_loss: 0.1041
Epoch 14/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1056 - val_loss: 0.1046
Epoch 15/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1049 - val_loss: 0.1029
Epoch 16/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1048 - val_loss: 0.1074
Epoch 17/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 20ms/step - loss: 0.1051 - val_loss: 0.1043
Epoch 18/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 10s 18ms/step - loss: 0.1040 - val_loss: 0.1069
Epoch 19/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1040 - val_loss: 0.1069
Epoch 20/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1043 - val_loss: 0.1040
Epoch 21/50
```

```
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1046 - val_loss: 0.1054
Epoch 22/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1047 - val_loss: 0.1041
Epoch 23/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.1058 - val_loss: 0.1043
Epoch 24/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1036 - val_loss: 0.1056
Epoch 25/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1037 - val_loss: 0.1048
Epoch 26/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1050 - val_loss: 0.1058
Epoch 27/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1023 - val_loss: 0.1036
Epoch 28/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 11s 18ms/step - loss: 0.1038 - val_loss: 0.1036
Epoch 29/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 10s 16ms/step - loss: 0.1014 - val_loss: 0.1023
Epoch 30/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1035 - val_loss: 0.1026
Epoch 31/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - loss: 0.1032 - val_loss: 0.1035
Epoch 32/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1035 - val_loss: 0.1037
Epoch 33/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.1039 - val_loss: 0.1038
Epoch 34/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.1018 - val_loss: 0.1052
Epoch 35/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - loss: 0.1033 - val_loss: 0.1077
Epoch 36/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1016 - val_loss: 0.1026
Epoch 37/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1025 - val_loss: 0.1042
Epoch 38/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - loss: 0.1014 - val_loss: 0.1053
Epoch 39/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1020 - val_loss: 0.1046
Epoch 40/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1009 - val_loss: 0.1067
Epoch 41/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1041 - val_loss: 0.1044
```

```
Epoch 42/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1018 - val_loss: 0.1053
Epoch 43/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.1005 - val_loss: 0.1034
Epoch 44/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1020 - val_loss: 0.1055
Epoch 45/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1015 - val_loss: 0.1004
Epoch 46/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 20ms/step - loss: 0.1003 - val_loss: 0.1052
Epoch 47/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1030 - val_loss: 0.1045
Epoch 48/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1022 - val_loss: 0.1040
Epoch 49/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 10s 17ms/step - loss: 0.0983 - val_loss: 0.1027
Epoch 50/50
328/328 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 0.0987 - val_loss: 0.1047
92/92 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step
Mean Absolute Error: 230455.04662356785
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
Next 7 days forecast: [394895.16 228965.2  176321.48 236056.75 304060.1  299512.25 183971.39]
```

In [ ]:

```
#Code for ARIMA
```

In [2]:

```
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error

# Load data
data = pd.read_csv('Dispense.csv', parse_dates=['caldate'])
data.set_index('caldate', inplace=True)

# Train ARIMA model
```

```
train = data['Dispense'][:int(0.8 * len(data))]
test = data['Dispense'][int(0.8 * len(data)):]

arima_model = ARIMA(train, order=(5, 1, 0))
arima_model_fit = arima_model.fit()

# Predict and evaluate
predictions = arima_model_fit.forecast(steps=len(test))
mae = mean_absolute_error(test, predictions)
print(f'Mean Absolute Error: {mae}')

# Forecast next 7 days
forecast = arima_model_fit.forecast(steps=7)
print('Next 7 days forecast:', forecast.values)
```

```
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided and will
be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided and will
be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided and will
be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
Mean Absolute Error: 226170.125146452
Next 7 days forecast: [435743.74359247 381136.10840052 420346.66724897 326334.07003046
 470775.18501113 341920.84405613 397809.55674381]
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in an exception.
  return get_prediction_index(
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
D:\New folder\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in an exception.
  return get_prediction_index(
```

Tried to Implement a short code here .....

The Results are as follows :

LSTM: Mean Absolute Error: 230455.04662356785, Next 7 days forecast: [394895.16 228965.2 176321.48 236056.75 304060.1 299512.25 183971.39]

ARIMA: Mean Absolute Error: 226170.125146452 , Next 7 days forecast: [435743.74359247 381136.10840052 420346.66724897 326334.07003046 470775.18501113 341920.84405613 397809.55674381]

Problem Solution Using LSTM .... by LIPIKA SHARMA

Data Preprocessing

In [58]:
```
# Import necessary libraries
```

In [59]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in d:\new folder\lib\site-packages (2.17.0)
Requirement already satisfied: tensorflow-intel==2.17.0 in d:\new folder\lib\site-packages (from tensorflow) (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow)
(2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow)
(1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorfl
ow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in d:\new folder\lib\site-packages (from tensorflow-intel==
2.17.0->tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflo
w) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (3.1
1.0)
Requirement already satisfied: libclang>=13.0.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow)
(18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tenso
rflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow)
(3.3.0)
Requirement already satisfied: packaging in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in d:\new folde
r\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflo
w) (2.31.0)
Requirement already satisfied: setuptools in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (68.2.
2)
Requirement already satisfied: six>=1.12.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (1.1
6.0)
Requirement already satisfied: termcolor>=1.1.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow)
(2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tens
orflow) (4.9.0)
Requirement already satisfied: wrapt>=1.11.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (1.
14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflo
w) (1.65.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tenso
rflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (3.
4.1)
```

```
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in d:\new folder\lib\site-packages (from tensorflow-intel==
2.17.0->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in d:\new folder\lib\site-packages (from tensorflow-intel==2.17.0->tensorfl
ow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in d:\new folder\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel=
=2.17.0->tensorflow) (0.41.2)
Requirement already satisfied: rich in d:\new folder\lib\site-packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflo
w) (13.3.5)
Requirement already satisfied: namex in d:\new folder\lib\site-packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflo
w) (0.0.8)
Requirement already satisfied: optree in d:\new folder\lib\site-packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorfl
ow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\new folder\lib\site-packages (from requests<3,>=2.21.0->tensorflo
w-intel==2.17.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in d:\new folder\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.1
7.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\new folder\lib\site-packages (from requests<3,>=2.21.0->tensorflow-inte
l==2.17.0->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in d:\new folder\lib\site-packages (from requests<3,>=2.21.0->tensorflow-inte
l==2.17.0->tensorflow) (2024.6.2)
Requirement already satisfied: markdown>=2.6.8 in d:\new folder\lib\site-packages (from tensorboard<2.18,>=2.17->tensorflow-int
el==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in d:\new folder\lib\site-packages (from tensorboard<2.18,
>=2.17->tensorflow-intel==2.17.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in d:\new folder\lib\site-packages (from tensorboard<2.18,>=2.17->tensorflow-int
el==2.17.0->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in d:\new folder\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.18,>=
2.17->tensorflow-intel==2.17.0->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in d:\new folder\lib\site-packages (from rich->keras>=3.2.0->tensor
flow-intel==2.17.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in d:\new folder\lib\site-packages (from rich->keras>=3.2.0->tensorflow-
intel==2.17.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in d:\new folder\lib\site-packages (from markdown-it-py<3.0.0,>=2.2.0->rich->keras>=
3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.1.0)
```

In [60]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

In [61]:
```
# Load the data
data = pd.read_csv('Dispense.csv')
data.head()
```

Out[61]:

	Account	ATMID	caldate	Dispense	DT	MaxCapacity	CountTotalTxn
0	ABC	SPCN02020	01-01-2021	564500	0	2640000	157
1	ABC	TPCN10269	01-01-2021	509000	9	3520000	92
2	ABC	APCN00816	01-01-2021	64800	0	2640000	36
3	PQR	S1CN1142	01-01-2021	834500	0	3520000	101
4	PQR	S1CN2022	01-01-2021	825700	0	2860000	364

In [62]:
```
# Convert caldate to datetime
data['caldate'] = pd.to_datetime(data['caldate'], format='%d-%m-%Y')
data = data.sort_values(by=['ATMID', 'caldate'])
```

Exploratory Data Analysis (EDA)

In [63]:
```
# Summary statistics
data.describe()
```

Out[63]:

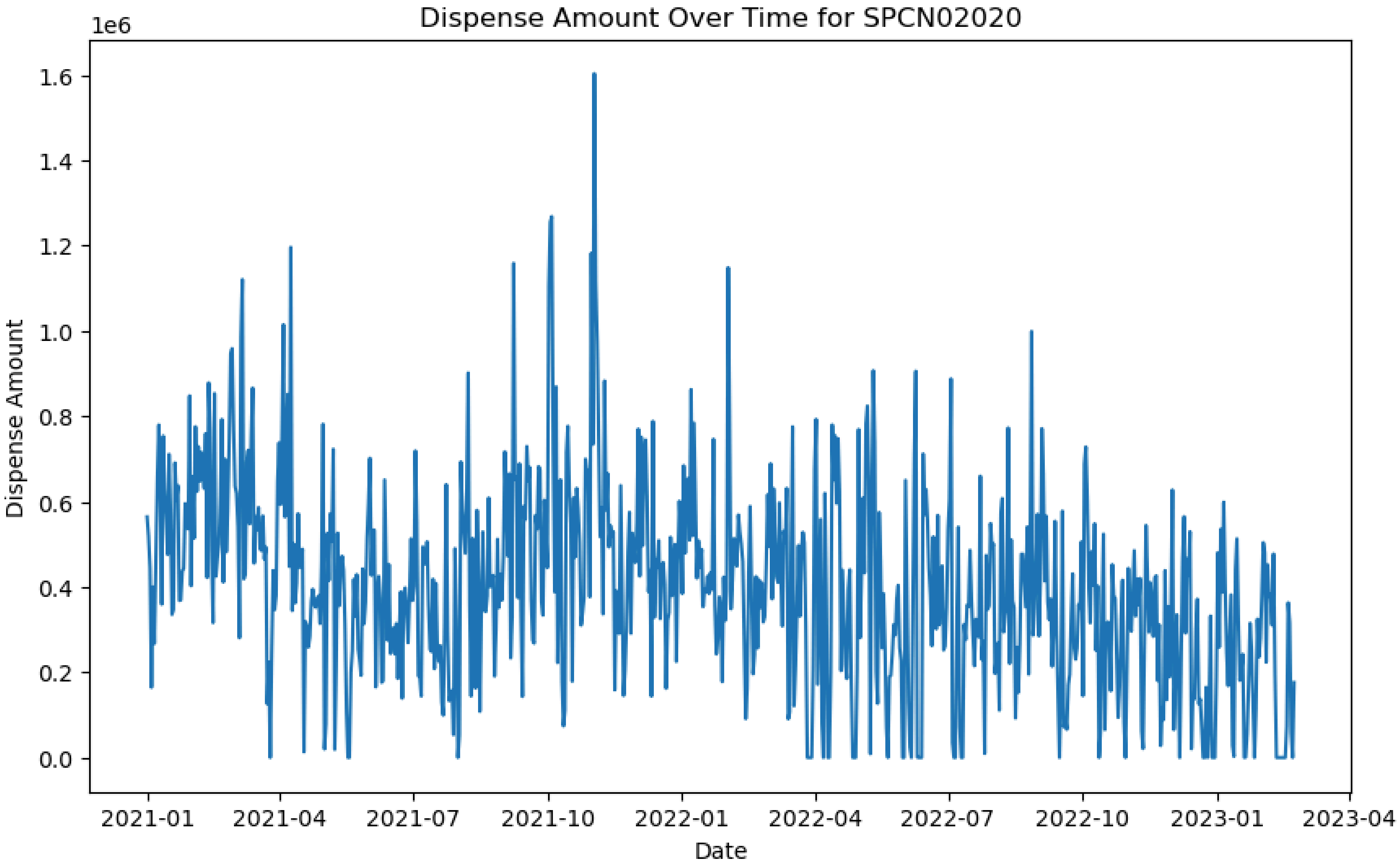
	caldate	Dispense	DT	MaxCapacity	CountTotalTxn
count	14593	1.459300e+04	14593.000000	1.459300e+04	14593.000000
mean	2022-02-24 13:07:26.762146304	4.027993e+05	158.538614	3.279753e+06	105.322963
min	2021-01-01 00:00:00	0.000000e+00	0.000000	2.420000e+06	0.000000
25%	2021-08-28 00:00:00	1.685000e+05	0.000000	2.860000e+06	45.000000
50%	2022-03-07 00:00:00	3.653000e+05	0.000000	3.520000e+06	98.000000
75%	2022-09-02 00:00:00	5.780000e+05	64.000000	3.520000e+06	146.000000
max	2023-02-22 00:00:00	2.151800e+06	1440.000000	3.740000e+06	561.000000
std	NaN	3.036762e+05	356.073765	3.974770e+05	76.727151

In [66]:

```
# Handle missing values
data.ffill(inplace=True)
```

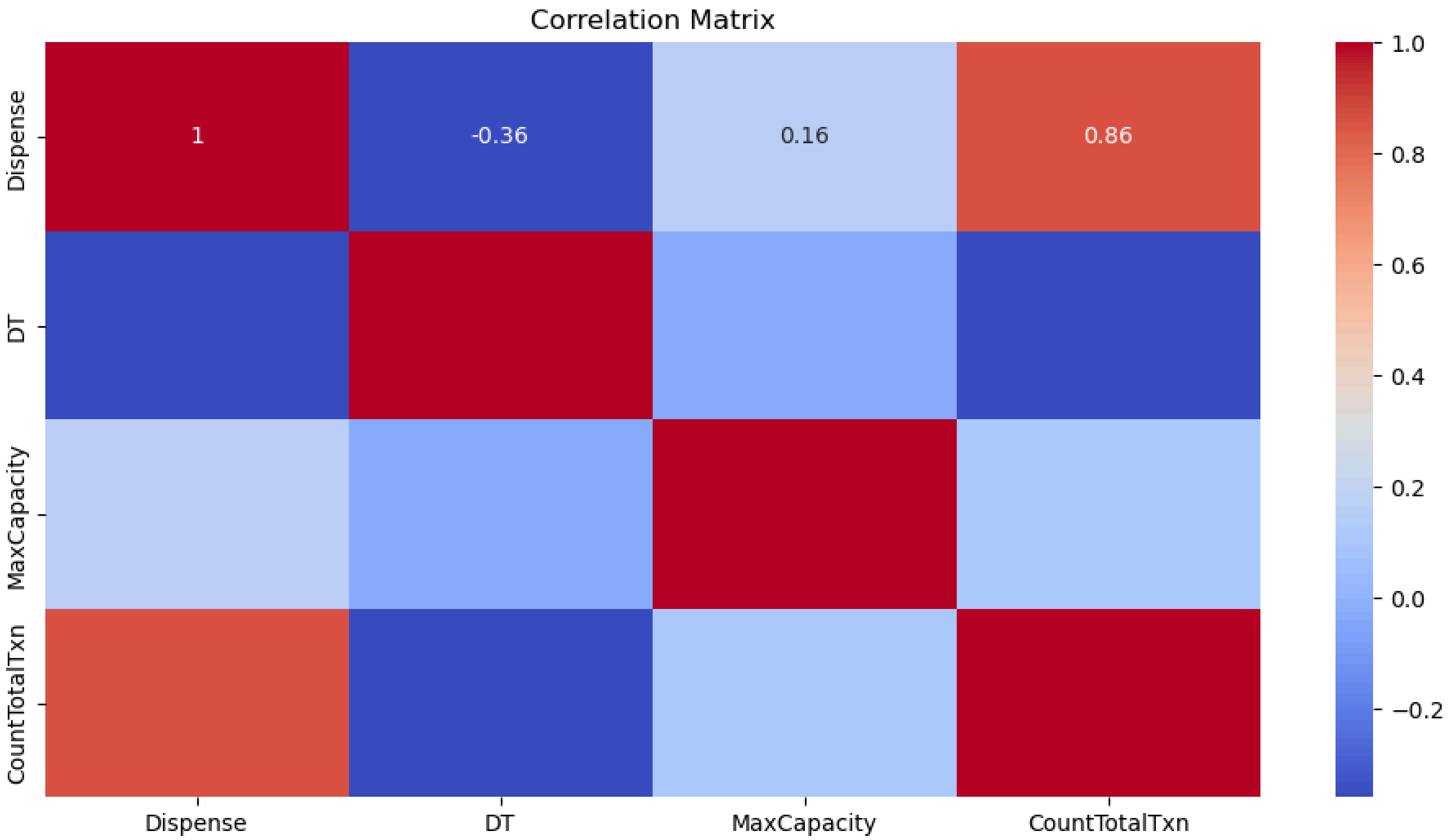
In [67]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Dispense amount over time for a sample ATM
sample_atm = data[data['ATMID'] == 'SPCN02020']
plt.figure(figsize=(10, 6))
plt.plot(sample_atm['caldate'], sample_atm['Dispense'])
plt.title('Dispense Amount Over Time for SPCN02020')
plt.xlabel('Date')
plt.ylabel('Dispense Amount')
plt.show()
```



In [68]:
```
# Select only numeric columns
numeric_data = data.select_dtypes(include=[np.number])

# Correlation matrix
plt.figure(figsize=(12, 6))
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Matrix')
plt.show()
```



Feature Engineering

In [69]:
```
def create_features(df):
    df['year'] = df['caldate'].dt.year
```

```
    df['month'] = df['caldate'].dt.month
    df['day'] = df['caldate'].dt.day
    df['weekday'] = df['caldate'].dt.weekday
    return df

data = create_features(data)

# Normalize the dispense amount
scaler = StandardScaler()
data['Dispense_scaled'] = scaler.fit_transform(data[['Dispense']])
```

Model Selection and Training (LSTM Approach)

In [70]:
```
# Creating sequences for LSTM
def create_sequences(df, seq_length):
    sequences = []
    for i in range(len(df) - seq_length):
        sequence = df['Dispense_scaled'].values[i:i+seq_length]
        label = df['Dispense_scaled'].values[i+seq_length]
        sequences.append((sequence, label))
    return sequences

seq_length = 30  # Using the past 30 days to predict the next day
atm_sequences = {}

for atm_id in data['ATMID'].unique():
    atm_data = data[data['ATMID'] == atm_id]
    atm_sequences[atm_id] = create_sequences(atm_data, seq_length)

# Preparing the data for LSTM
X, y = [], []
for atm_id in atm_sequences:
    atm_X, atm_y = zip(*atm_sequences[atm_id])
    X.extend(atm_X)
    y.extend(atm_y)

X = np.array(X)
y = np.array(y)
```

In [71]:
```
# Split into train and test sets
split_idx = int(0.8 * len(X))
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

# Reshape input to be 3D for LSTM [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(seq_length, 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

```
D:\New folder\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 30, 50)	10,400
lstm_5 (LSTM)	(None, 50)	20,200
dropout_2 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 1)	51

Total params: 30,651 (119.73 KB)

Trainable params: 30,651 (119.73 KB)

Non-trainable params: 0 (0.00 B)

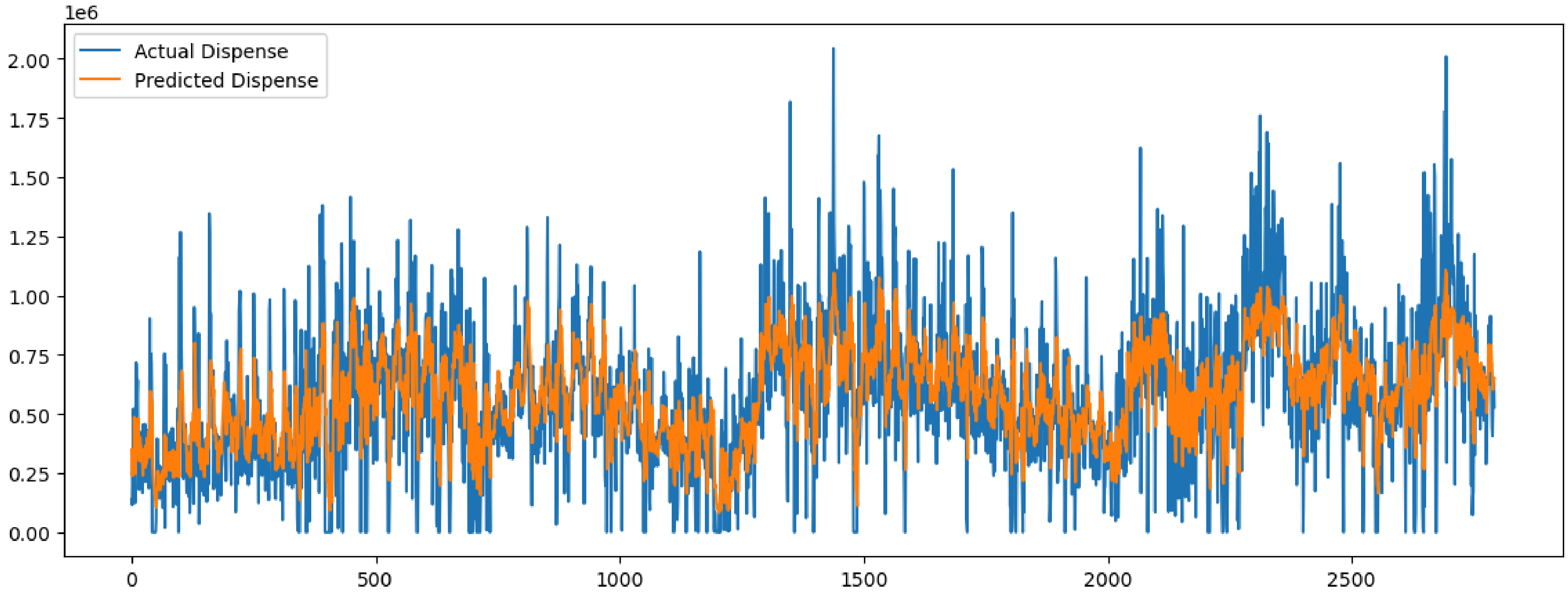
In [72]:
```
# Train the model
early_stop = EarlyStopping(monitor='val_loss', patience=5)
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2, callbacks=[early_stop])

# Evaluate the model
y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred)
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))

mse = mean_squared_error(y_test_inv, y_pred_inv)
print(f'Mean Squared Error: {mse}')
```

```
Epoch 1/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 12s 22ms/step - loss: 0.5938 - val_loss: 0.3643
Epoch 2/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - loss: 0.5053 - val_loss: 0.3531
Epoch 3/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - loss: 0.4685 - val_loss: 0.3384
Epoch 4/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 10s 18ms/step - loss: 0.4781 - val_loss: 0.3365
Epoch 5/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.4486 - val_loss: 0.3303
Epoch 6/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - loss: 0.4596 - val_loss: 0.3371
Epoch 7/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - loss: 0.4813 - val_loss: 0.3304
Epoch 8/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 6s 21ms/step - loss: 0.4771 - val_loss: 0.3401
Epoch 9/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - loss: 0.4696 - val_loss: 0.3382
Epoch 10/50
280/280 ━━━━━━━━━━━━━━━━━━━━ 10s 18ms/step - loss: 0.4565 - val_loss: 0.3333
88/88 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
Mean Squared Error: 60678255268.89817
```

Visualization

In [73]:
```
# Visualization of actual vs predicted
plt.figure(figsize=(14, 5))
```

```
plt.plot(y_test_inv, label='Actual Dispense')
plt.plot(y_pred_inv, label='Predicted Dispense')
plt.legend()
plt.show()
```



Forecasting

In [75]:

```
# Forecasting next 7 days
def forecast_next_days(model, data, seq_length, n_days):
    forecasts = []
    input_seq = data.reshape((1, seq_length, 1))  # Reshape to match the input format of the model
    for _ in range(n_days):
        next_dispense = model.predict(input_seq)
        next_dispense = next_dispense.reshape((1, 1, 1))  # Reshape next_dispense to match the dimensions
        forecasts.append(next_dispense[0, 0, 0])
        input_seq = np.append(input_seq[:, 1:, :], next_dispense, axis=1)
    return forecasts
```

```
# Forecast for each ATM
forecast_results = {}
for atm_id in data['ATMID'].unique():
    atm_data = data[data['ATMID'] == atm_id]
    atm_seq = atm_data['Dispense_scaled'].values[-seq_length:]
    forecast = forecast_next_days(model, atm_seq, seq_length, 7)
    forecast_inv = scaler.inverse_transform(np.array(forecast).reshape(-1, 1))
    forecast_results[atm_id] = forecast_inv
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
```

In [76]:

```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

In [77]:

```python
# Assuming `data` is your DataFrame and `model` is your trained model

# Define forecast_next_days function
def forecast_next_days(model, atm_seq, seq_length, forecast_days):
    # Dummy implementation for illustration; replace with your actual function
    return [0.5] * forecast_days

# Initialize and fit the MinMaxScaler on the original 'Dispense_scaled' column
scaler = MinMaxScaler()
scaler.fit(data['Dispense_scaled'].values.reshape(-1, 1))
```

```
# Length of the sequence used for prediction
seq_length = 30
```

In [78]:

```
# Forecast for each ATM
forecast_results = {}
for atm_id in data['ATMID'].unique():
    atm_data = data[data['ATMID'] == atm_id]

    # Ensure there are enough data points to form a sequence
    if len(atm_data) >= seq_length:
        atm_seq = atm_data['Dispense_scaled'].values[-seq_length:]

        # Check the shape of atm_seq
        if atm_seq.shape[0] == seq_length:
            forecast = forecast_next_days(model, atm_seq, seq_length, 7)

            # Ensure forecast is a numpy array
            forecast = np.array(forecast).reshape(-1, 1)

            # Check if forecast can be inverse transformed
            if forecast.shape[1] == 1:
                forecast_inv = scaler.inverse_transform(forecast)
                forecast_results[atm_id] = forecast_inv
            else:
                print(f"Error: Forecast shape {forecast.shape} is not suitable for inverse transform")
        else:
            print(f"Error: ATM {atm_id} sequence length {atm_seq.shape[0]} does not match {seq_length}")
    else:
        print(f"Error: Not enough data for ATM {atm_id}")

# Check the forecast results
print(forecast_results)
```

```
{'APAN11109': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416]]), 'APAN22403': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416]]), 'APAN23217': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416]]), 'APAN35706': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416]]), 'APCN00816': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416]]), 'S1CN1142': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416]]), 'S1CN2011': array([[2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
       [2.21658416],
```

```
[2.21658416],
[2.21658416]]), 'S1CN2022': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'S1CN2820': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'S1CN3514': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'SPCN02020': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'SPCND067': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'SPCNG376': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'T1BH000003039': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
```

[2.21658416],
[2.21658416],
[2.21658416]]), 'T1BH000011116': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'T1BH000274012': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'T1BH000603091': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'T1BH007252090': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'T1NH000575414': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'T1NY000166081': array([[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]]), 'TPCN10269': array([[2.21658416],
[2.21658416],
[2.21658416],

```
[2.21658416],
[2.21658416],
[2.21658416],
[2.21658416]])}
```

Save the Final Result

In [79]:

```
import numpy as np
import pandas as pd

# Assuming forecast_results is a dictionary with ATM IDs as keys and 2D forecast arrays as values
flattened_results = {atm_id: forecast.flatten() for atm_id, forecast in forecast_results.items()}

# Convert the flattened results dictionary to a DataFrame
forecast_df = pd.DataFrame.from_dict(flattened_results, orient='index')
forecast_df.columns = [f'Day_{i+1}' for i in range(forecast_df.shape[1])]

# Save the DataFrame to a CSV file
forecast_df.to_csv('atm_dispense_forecasts.csv', index=True)

# Save the Jupyter Notebook and Python script
# (In Jupyter Notebook, use the following code to save the notebook)
# !jupyter nbconvert --to script atm_dispense_forecast.ipynb

# Save the model in the native Keras format
model.save('atm_dispense_model.keras')
```