Task 3

Market Price PREDICTION

Ibrahim Tarek Abdelazeem

Table of CONTENTS

Data Preprocessing

Loading Data

- Read data from a CSV file into a DataFrame.

Handling Missing Values

- Impute missing values using the mean.
- Columns: quantity, priceMin, priceMax, priceMod.

Encoding Categorical Variables

- Convert categorical variables into numerical format.
- Variables: market, state, city.

Extracting Date Components

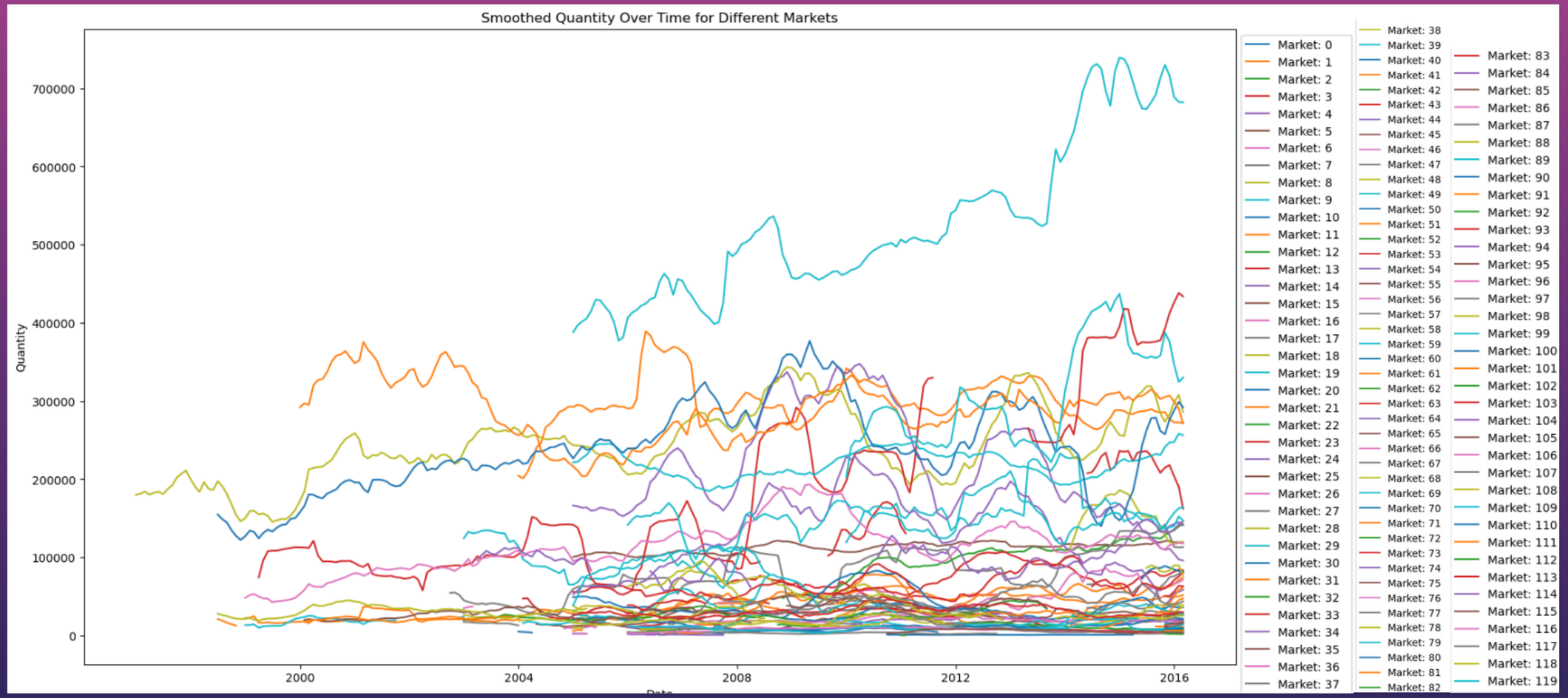
- Convert date column to datetime format.
- Extract year and month from date.

Data Preprocessing

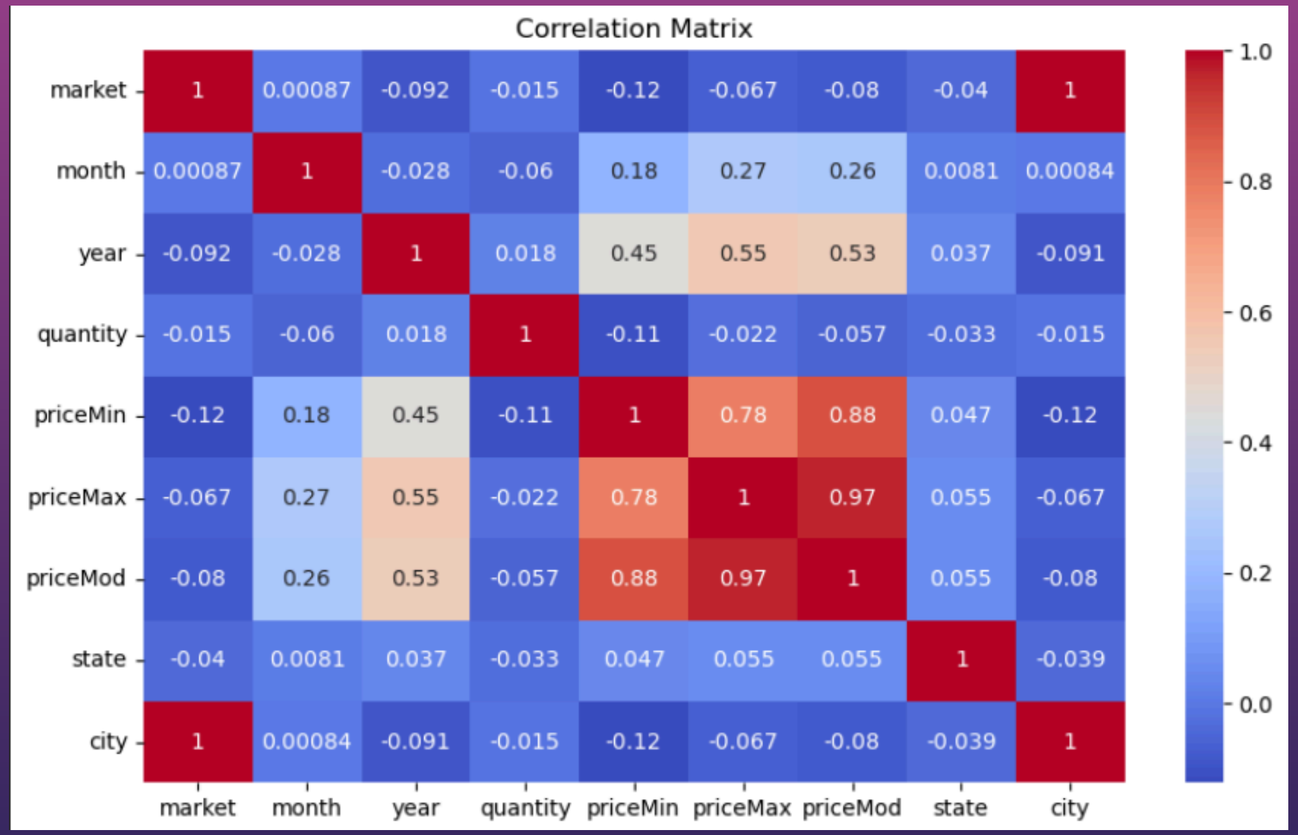
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer

# Load data
df = pd.read_csv('MarketPricePrediction.csv')

# Handle missing values
imputer = SimpleImputer(strategy='mean')
df['quantity'] = imputer.fit_transform(df[['quantity']])
df['priceMin'] = imputer.fit_transform(df[['priceMin']])
df['priceMax'] = imputer.fit_transform(df[['priceMax']])
df['priceMod'] = imputer.fit_transform(df[['priceMod']])

# Encode categorical variables
le_market = LabelEncoder()
df['market'] = le_market.fit_transform(df['market'])
le_state = LabelEncoder()
df['state'] = le_state.fit_transform(df['state'])
le_city = LabelEncoder()
df['city'] = le_city.fit_transform(df['city'])

# Extract date components
df['date'] = pd.to_datetime(df['date'])
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
print(df.head())
```

```
...    market  month  year  quantity  priceMin  priceMax  priceMod  state  city  \
0       0      1  2005    2350.0     404.0     493.0     446.0     16     0
1       0      1  2006     900.0     487.0     638.0     563.0     16     0
2       0      1  2010     790.0    1283.0    1592.0    1460.0     16     0
3       0      1  2011     245.0    3067.0    3750.0    3433.0     16     0
4       0      1  2012    1035.0     523.0     686.0     605.0     16     0

        date
0 2005-01-01
1 2006-01-01
2 2010-01-01
3 2011-01-01
4 2012-01-01
```

Exploratory Data Analysis



Exploratory Data Analysis



Feature Engineering

Lagged Features

- Capture temporal dependencies and autocorrelation.
- Create lagged versions of quantity.

Rolling Statistics

- Capture local trends and variability.
- Create rolling mean and standard deviation features.

Seasonal Indicators

- Capture seasonal patterns.
- Create dummy variables for months.

Dropping NaNs

- Remove rows with NaN values created by shifting operations.

Feature Engineering

```python
# Create lagged features
df['quantity_lag1'] = df['quantity'].shift(1)
df['quantity_lag2'] = df['quantity'].shift(2)

# Create rolling statistics
df['quantity_roll_mean'] = df['quantity'].rolling(window=3).mean()
df['quantity_roll_std'] = df['quantity'].rolling(window=3).std()

# Create seasonal indicators
df = pd.get_dummies(df, columns=['month'], drop_first=True)

# Drop rows with NaN values created by shifting
df = df.dropna()
print(df.head())
```

Model Selection and Training

Models Chosen for Forecasting

- **SARIMA (Seasonal Autoregressive Integrated Moving Average)**
- **LSTM (Long Short-Term Memory)**

Why SARIMA?

Strengths of SARIMA

- Captures Seasonality:
  - Well-suited for data with strong seasonal patterns.
  - Incorporates seasonal differencing to handle seasonality directly.
- Interpretable Parameters:
  - Easy to understand and interpret model parameters.
  - Good for explaining the influence of past values and errors on current values.

Why Chose SARIMA?

- Data Characteristics:
  - The dataset showed evidence of seasonal trends.
  - SARIMA's ability to handle seasonality made it a suitable choice.

Why LSTM?

Strengths of LSTM

- Handles Complex Patterns:
  - Can capture long-term dependencies and complex temporal patterns.
  - Suitable for data with non-linear relationships.
- Sequential Data Processing:
  - Designed for sequential data, making it ideal for time series.
  - Uses memory cells to retain information over time.

Why Chose LSTM?

- *A*dvanced Modeling Capability:
  - LSTM's ability to model complex temporal relationships and non-linearities.
  - Powerful tool for capturing intricate patterns in the data.

Evaluation Metrics

Metrics Used

- **Mean Absolute Error (MAE):**
  - **Measures the average absolute difference between predicted and actual values.**
- **Mean Squared Error (MSE):**
  - **Measures the average squared difference, penalizing larger errors more heavily.**
- **Root Mean Squared Error (RMSE):**
  - **Square root of MSE, providing error in the same units as the target variable.**

Which Model is Better?

Model Evaluation

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Evaluate SARIMA
sarima_pred = forecast.predicted_mean
mae_sarima = mean_absolute_error(test['quantity'], sarima_pred)
mse_sarima = mean_squared_error(test['quantity'], sarima_pred)
rmse_sarima = np.sqrt(mse_sarima)

# Evaluate LSTM
mae_lstm = mean_absolute_error(scaler.inverse_transform(y_test.reshape(-1, 1)), predictions)
mse_lstm = mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)), predictions)
rmse_lstm = np.sqrt(mse_lstm)

print(f'SARIMA - MAE: {mae_sarima}, MSE: {mse_sarima}, RMSE: {rmse_sarima}')
print(f'LSTM - MAE: {mae_lstm}, MSE: {mse_lstm}, RMSE: {rmse_lstm}')
```

27] ✓ 0.0s

```
SARIMA - MAE: 52889.471534011725, MSE: 10134598865.415205, RMSE: 100670.74483391491
LSTM - MAE: 30677.19255632801, MSE: 2955460539.6785936, RMSE: 54364.147557729564
```

Which Model is Better?

SARIMA Performance

- **MAE: 52,889.47**
- **MSE: 10,134,598,865.42**
- **RMSE: 100,670.74**

LSTM Performance

- **MAE: 30,677.19**
- **MSE: 2,955,460,539.68**
- **RMSE: 54,364.15**

Which Model is Better?

Comparing Performance

- **Mean Absolute Error (MAE):**
  - LSTM (30,677.19) vs. SARIMA (52,889.47)
  - Lower MAE for LSTM indicates better average accuracy.
- **Mean Squared Error (MSE):**
  - LSTM (2,955,460,539.68) vs. SARIMA (10,134,598,865.42)
  - Lower MSE for LSTM shows fewer large errors.
- **Root Mean Squared Error (RMSE):**
  - LSTM (54,364.15) vs. SARIMA (100,670.74)
  - Lower RMSE for LSTM indicates better overall performance.

Conclusion

- **LSTM is the better performing model based on all three metrics.**