Sales Forecasting Models

June 14, 2024

0.0.1 1. Data Processing

[1]:

```
import pandas as pd
import numpy as np
```

[2]:

```
#Load Data
df=pd.read_excel(r"C:\Users\USER\Documents\Data Portfolio Projects\Retail\Sales␣
 ↪Forecasting\refined_sales_forecasting_dataset.xlsx")
df.head()
```

[2]:

```
        Date  Langa_Fresh Produce  Langa_Dairy  Langa_Canned Goods  \
0 2021-01-01                  NaN          NaN                 NaN
1 2021-01-02                 15.0         20.0                21.0
2 2021-01-03                 21.0         28.0                20.0
3 2021-01-04                 25.0         22.0                16.0
4 2021-01-05                 15.0         17.0                17.0

   Langa_Bakery  Langa_Frozen Foods  Nyanga_Fresh Produce  Nyanga_Dairy  \
0           NaN                 NaN                   NaN           NaN
1           9.0                29.0                  16.0          18.0
2          17.0                20.0                  17.0          18.0
3          20.0                22.0                  29.0          21.0
4          26.0                17.0                  16.0          11.0

   Nyanga_Canned Goods  Nyanga_Bakery  …  Pinelands_Bakery  \
0                  NaN            NaN  …               NaN
1                 18.0           21.0  …              17.0
2                 17.0           20.0  …              17.0
3                 17.0           20.0  …              18.0
4                 27.0           22.0  …              22.0

   Pinelands_Frozen Foods  Thornton_Fresh Produce  Thornton_Dairy  \
0                     NaN                     NaN             NaN
1                    26.0                    18.0            11.0
2                    25.0                    26.0            18.0
3                    33.0                    18.0            20.0
4                    23.0                    22.0            30.0
```

```
   Thornton_Canned Goods  Thornton_Bakery  Thornton_Frozen Foods  \
0                    NaN              NaN                    NaN
1                   20.0             20.0                   16.0
2                   25.0             16.0                   23.0
3                   18.0             17.0                   14.0
4                   23.0             23.0                   26.0

   Economic_Indicator  Holiday_Indicator  Promotion_Indicator
0           88.728706                  0                    0
1          115.513937                  0                    0
2           87.385129                  0                    0
3          114.177901                  0                    0
4           96.823213                  0                    0

[5 rows x 29 columns]
```

[3]:

```
# Handle missing values by imputing with the median
df.fillna(df.median(), inplace=True)
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_13268\32931963.py:2: FutureWarning:
DataFrame.mean and DataFrame.median with numeric_only=None will include
datetime64 and datetime64tz columns in a future version.
  df.fillna(df.median(), inplace=True)
```

[4]:

```
#Create additional features that may help in forecasting, such as day of the␣
 ↪week, month, and year

df['Day_of_Week'] = df['Date'].apply(lambda x: pd.Timestamp(x).dayofweek)
df['Month'] = df['Date'].apply(lambda x: pd.Timestamp(x).month)
df['Year'] = df['Date'].apply(lambda x: pd.Timestamp(x).year)
```

[ ]:

```

```

0.0.2 Exploratory Data Analysis

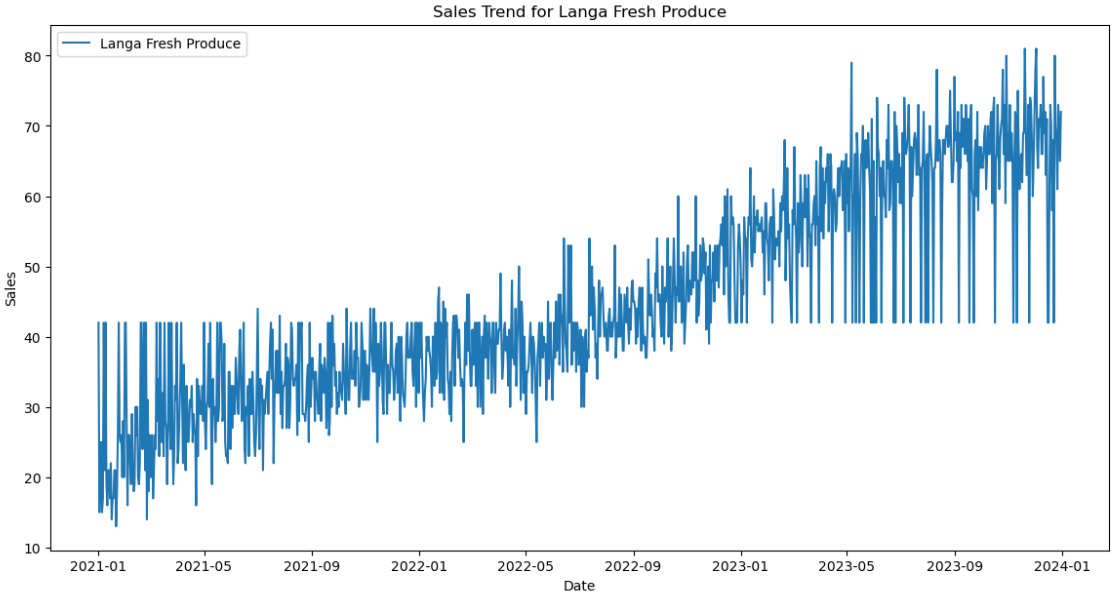
Data will be visualized to identify trends, seasonality, and relationships between variables.

[5]:

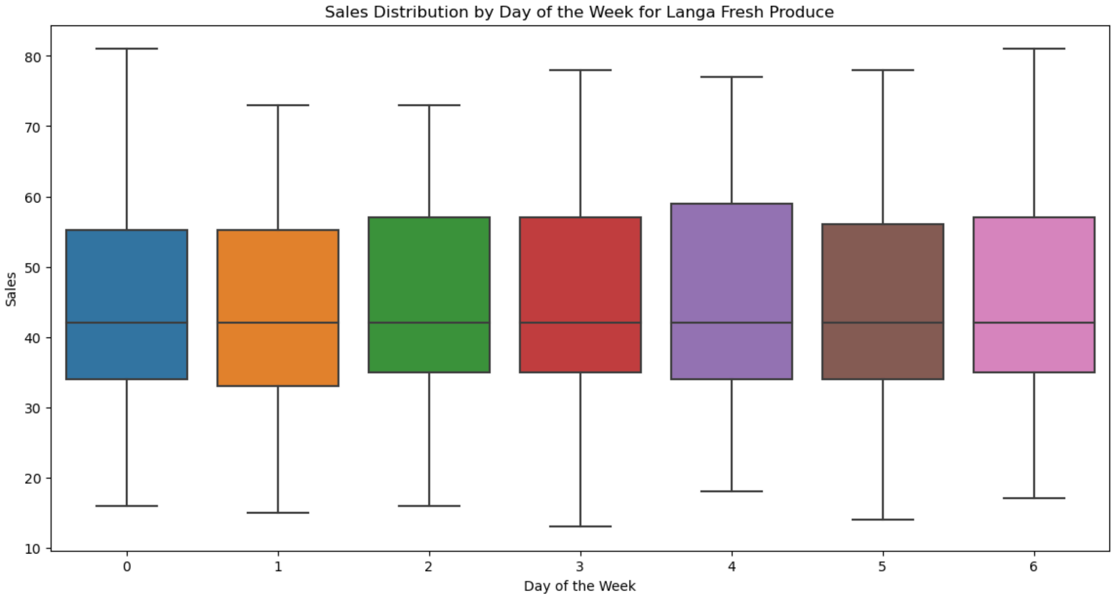
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[6]:

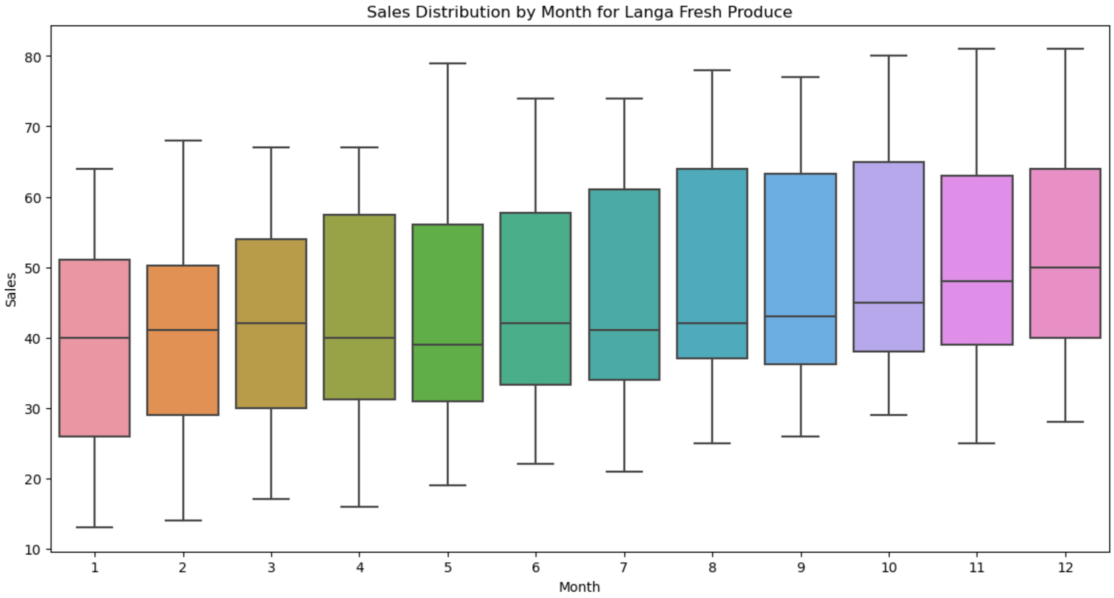
```
# Plot of the sales trends for the Langa Store and the Fresh Produce category
plt.figure(figsize=(14, 7))
plt.plot(df['Date'], df['Langa_Fresh Produce'], label='Langa Fresh Produce')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales Trend for Langa Fresh Produce')
plt.legend()
plt.show()
```


Sales Trend for Langa Fresh Produce

[7]:
```python
# Sales by day of the week
plt.figure(figsize=(14, 7))
sns.boxplot(x='Day_of_Week', y='Langa_Fresh Produce', data=df)
plt.xlabel('Day of the Week')
plt.ylabel('Sales')
plt.title('Sales Distribution by Day of the Week for Langa Fresh Produce')
plt.show()
```


Sales Distribution by Day of the Week for Langa Fresh Produce

[8]:
```
# Sales by month
plt.figure(figsize=(14, 7))
sns.boxplot(x='Month', y='Langa_Fresh Produce', data=df)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales Distribution by Month for Langa Fresh Produce')
plt.show()
```



[ ]:
```

```

0.0.3 Model Selection and Training

We'll use a simple ARIMA model for time series forecasting. We'll start by preparing the data for the ARIMA model and then train the model.

Preparing the Data

[9]:
```
from statsmodels.tsa.arima.model import ARIMA
```

[10]:
```
# Convert the date column to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

[11]:
```
# Set the date column as the index
df.set_index('Date', inplace=True)
```

[12]:
```
# Select the time series for the Langa store and Fresh Produce category
series = df['Langa_Fresh Produce']
```

[13]:

```
# Split the data into training and testing sets
train_size = int(len(series) * 0.8)
train, test = series[:train_size], series[train_size:]
```

[ ]:

```

```

Train the ARIMA Model

[14]:

```
# Train the ARIMA model
model = ARIMA(train, order=(5, 1, 0))
model_fit = model.fit()

# Print the summary of the model
print(model_fit.summary())
```

```
C:\Users\USER\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
C:\Users\USER\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
C:\Users\USER\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
```

```
                               SARIMAX Results
================================================================================
Dep. Variable:     Langa_Fresh Produce   No. Observations:                  876
Model:                  ARIMA(5, 1, 0)   Log Likelihood               -2836.816
Date:                 Mon, 10 Jun 2024   AIC                           5685.632
Time:                         17:00:34   BIC                           5714.278
Sample:                     01-01-2021   HQIC                          5696.590
                          - 05-26-2023
Covariance Type:                   opg
===============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
ar.L1         -0.8381      0.031    -26.893      0.000      -0.899      -0.777
ar.L2         -0.6752      0.040    -16.746      0.000      -0.754      -0.596
ar.L3         -0.5233      0.042    -12.553      0.000      -0.605      -0.442
ar.L4         -0.3285      0.039     -8.496      0.000      -0.404      -0.253
ar.L5         -0.1719      0.030     -5.682      0.000      -0.231      -0.113
sigma2        38.2859      1.620     23.640      0.000      35.112      41.460
================================================================================
===
Ljung-Box (L1) (Q):                   0.09   Jarque-Bera (JB):
```

```
16.45
Prob(Q):                              0.76   Prob(JB):
0.00
Heteroskedasticity (H):               0.81   Skew:
-0.03
Prob(H) (two-sided):                  0.07   Kurtosis:
3.67
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

[ ]:

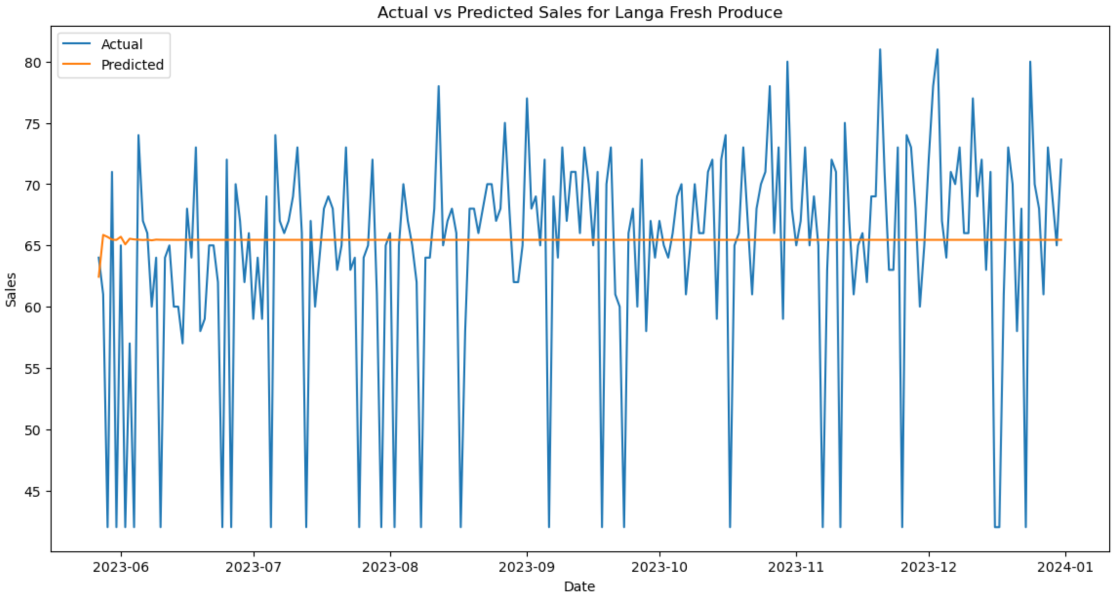
0.0.4 Model Evaluation

[15]:

```python
# Make predictions
predictions = model_fit.forecast(steps=len(test))
```

[16]:

```python
# Plot the actual vs predicted values
plt.figure(figsize=(14, 7))
plt.plot(test.index, test, label='Actual')
plt.plot(test.index, predictions, label='Predicted')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Actual vs Predicted Sales for Langa Fresh Produce')
plt.legend()
plt.show()
```



[17]:

```
# Calculate evaluation metrics
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(test, predictions)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')
```

```
RMSE: 9.184774172849416
```

[ ]:

```

```

0.0.5 Forecasting

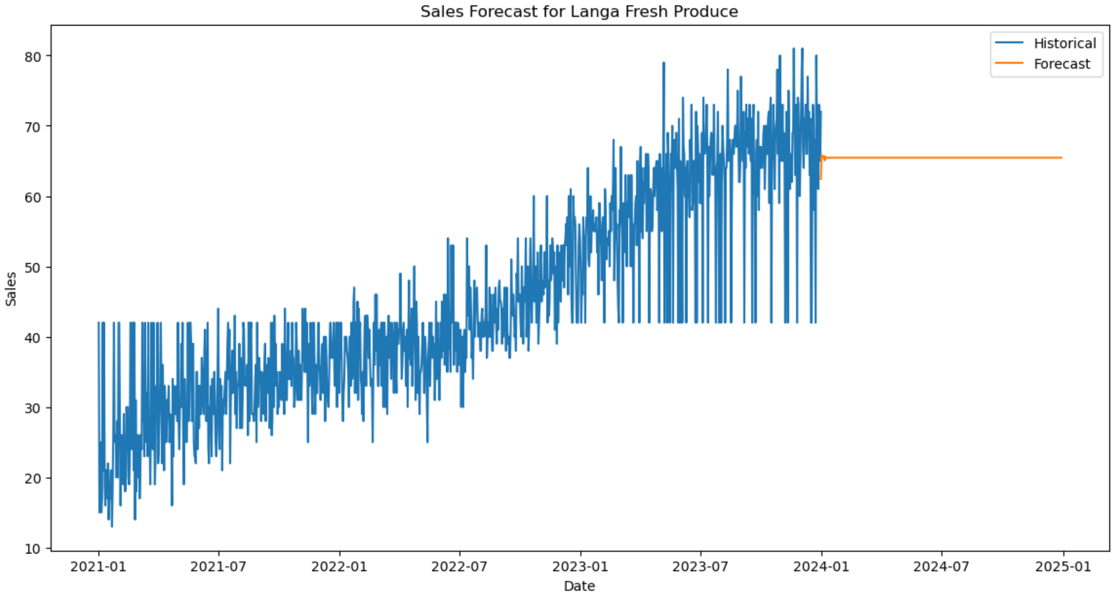
Finally, we'll generate forecasts for different time horizons (weekly, monthly, quarterly, annual) and visualize the results

[18]:

```
# Generate forecasts for the next year
future_steps = 365
forecast = model_fit.forecast(steps=future_steps)
```

[19]:

```
# Plot the forecast
plt.figure(figsize=(14, 7))
plt.plot(series.index, series, label='Historical')
plt.plot(pd.date_range(start=series.index[-1], periods=future_steps, freq='D'),␣
  ↪forecast, label='Forecast')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales Forecast for Langa Fresh Produce')
plt.legend()
plt.show()
```


Sales Forecast for Langa Fresh Produce
Historical
Forecast
80
70
60
50
40
30
20
10
Sales
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
2024-07
2025-01
Date

```
[ ]:
```