**Report**

|  |  |  |
| --- | --- | --- |
| **Initials** | **Surname** | **Student Number** |
| MN | Manqele | 22223185 |
| S | Mbawa | 22215849 |
| ST | Mdluli | 22203957 |
| NN | Nzama | 22314687 |
| OS | Xulu | 22222778 |

Comprehensive Report: Empowering Spaza Shops Through Time Series Forecasting

Project Theme:

Empowering Spaza Shops: Time Series Forecasting for Local Businesses in South Africa

**Introduction:**

Spaza shops are the backbone of South African communities, providing essential goods in urban and rural areas. These small, often informal businesses are family-run and play a crucial role in the local economy. However, they face challenges such as fluctuating demand, competition from larger retailers, and supply chain issues. Time series forecasting can help spaza shop owners predict business metrics like sales and inventory, helping them optimize operations and improve profitability.

This report details the process of developing a time series forecasting solution using three different models: Prophet, ARIMA, and LSTM. Each model was applied to predict the sales, revenue, and inventory levels of spaza shops. Data preparation, model implementation, and analysis steps are discussed in this report.

**Objectives:**

Data Collection & Preprocessing: Collect or create a dataset reflecting small business operations and preprocess the data for analysis.

Model Implementation: Use three models to forecast key business metrics:

Prophet

ARIMA

LSTM

Model Comparison: Compare the models to determine which one provides the most accurate predictions.

Insights & Recommendations: Based on model forecasts, provide actionable recommendations for spaza shop owners.

**Step 1: Data Collection and Preprocessing**

The dataset provided consists of spaza shop sales data, including:

* Date: The date of transaction or event.
* Product: The product sold.
* Sales (Units): Number of units sold.
* Revenue (ZAR): Total revenue generated.
* Inventory Levels: Stock available before the sale.
* Event Impact: Special events (e.g., festivals or holidays) affecting sales.
* Season: The season during the transaction.

**Dataset Example:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Product | Sales(units) | Revenue(ZAR) | Inventory Levels | Event Impact | Season |
| 2023-01-02 | Maize Meal | 25 | 875.50 | 35 | Local Festival | Summer |
| 2023-01-05 | Cooking Oil | 15 | 600.75 | 60 | Pubic Holiday | Summer |
| 2023-01-07 | Sugar | 10 | 300.30 | 50 | No Event | Summer |
| 2023-01-12 | Bread | 20 | 200.00 | 40 | No Evant | Summer |
| 2023-01-15 | Milk | 30 | 450.00 | 30 | No Event | Autumn |

**Data Preprocessing:**

**Missing Values:**

Missing values in the "Event Impact" column were filled with "No Event" as these represented non-event days.

**Python code**

data['Event Impact'].fillna('No Event', inplace=True)

Date Format Conversion:

The "Date" column was converted to datetime format to facilitate time series analysis.

data['Date'] = pd.to\_datetime(data['Date'])

Feature Encoding:

Categorical variables like "Season" and "Event Impact" were encoded using one-hot encoding.

data = pd.get\_dummies(data, columns=['Season', 'Event Impact'], drop\_first=True)

Exploratory Data Analysis (EDA):

A line plot was generated to visualize sales trends over time:

sns.lineplot(data=data, x='Date', y='Sales (Units)', marker='o')

plt.title('Sales Over Time')

plt.show()

**Step 2: Model Implementatio**n

**Model 1: Prophet**

Prophet is a time series forecasting model that handles seasonality, holidays, and trends automatically. It is well-suited for business forecasting.

**Data Preparation:**

The dataset was reformatted to meet Prophet's requirements:

prophet\_data = data[['Date', 'Sales (Units)']].rename(columns={'Date': 'ds', 'Sales (Units)': 'y'})

**Model Fitting:**

**code**

model\_prophet = Prophet()

model\_prophet.fit(prophet\_data)

Forecasting:

Predictions were made for 30 future periods, and the forecast was plotted:

**code**

future = model\_prophet.make\_future\_dataframe(periods=30)

forecast = model\_prophet.predict(future)

model\_prophet.plot(forecast)

Prophet Analysis:

Prophet automatically captured the seasonality and trend in the data, providing interpretable forecasts and plotting them with confidence intervals.

**Model 2: ARIMA**

The ARIMA (Auto-Regressive Integrated Moving Average) model is a widely used forecasting technique for time series data.

**Stationarity Check:**

Stationarity of the time series was checked using the Augmented Dickey-Fuller (ADF) test:

**code**

result = adfuller(data['Sales (Units)'])

Differencing:

If the data is non-stationary, first-order differencing was applied to make it stationary:

data['Sales Diff'] = data['Sales (Units)'].diff().dropna()

Model Fitting: An ARIMA model was fitted with appropriate order parameters:

arima\_model = ARIMA(data['Sales (Units)'], order=(1,1,1))

arima\_model\_fit = arima\_model.fit()

Forecasting: Predictions were made for the next 30 periods:

arima\_forecast = arima\_model\_fit.forecast(steps=30)

ARIMA Analysis:

ARIMA successfully modeled short-term dependencies in the sales data but struggled with long-term trends and seasonality compared to Prophet.

**Model 3: LSTM (Long Short-Term Memory)**

LSTM networks are a type of recurrent neural network capable of capturing long-term dependencies in time series data.

**Data Scaling:**

LSTM models require scaled data, so the sales data was scaled between 0 and 1 using MinMaxScaler:

**code**

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(data[['Sales (Units)']])

Data Reshaping:

Time steps of 10 were used to structure the data for LSTM input:

Python

X\_train, Y\_train = create\_dataset(train\_data, time\_step=10)

Model Building: A sequential LSTM model was built with layers for LSTM units, Dense layers for output, and dropout layers to prevent overfitting:

model\_lstm = Sequential()

model\_lstm.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)))

model\_lstm.add(LSTM(50, return\_sequences=False))

model\_lstm.add(Dense(25))

model\_lstm.add(Dense(1))

model\_lstm.compile(optimizer='adam', loss='mean\_squared\_error')

model\_lstm.fit(X\_train, Y\_train, batch\_size=1, epochs=1)

Forecasting: Predictions were generated for the test data, and the results were plotted:

python

predictions = model\_lstm.predict(X\_test)

predictions = scaler.inverse\_transform(predictions)

LSTM Analysis:

The LSTM model captured long-term dependencies in the data and performed well on sequences where ARIMA struggled. However, it required more computational resources and longer training time.

**Step 3: Model Comparison**

**Prophet:**

Strengths: Easy to implement, handles seasonality and holidays well.

Weaknesses: Struggles with sudden, unpredictable changes in data.

**ARIMA:**

Strengths: Simple and effective for short-term forecasts, especially when data is stationary.

Weaknesses: Doesn’t handle seasonality and trends as well as Prophet and LSTM.

**LSTM**:

Strengths: Excellent at capturing long-term patterns and dependencies in complex datasets.

Weaknesses: Requires large amounts of data and computational resources, can be prone to overfitting.

**Step 4: Insights & Recommendations**

**Based on the model outputs:**

**Stock Planning:**

Spaza shop owners should stock up on essential goods before local festivals or public holidays, as sales tend to spike during these periods (as seen from Prophet and ARIMA).

**Seasonality Adjustments:**

The LSTM model showed consistent seasonal demand patterns, indicating that shops should prepare for varying demand during different seasons.

**Inventory Management:**

Insights from Prophet and LSTM suggest that spaza shops can minimize waste by aligning inventory levels with predicted sales peaks during holidays and festivals.

**Conclusion**

This report demonstrated the application of time series forecasting models—Prophet, ARIMA, and LSTM—to spaza shop sales data. By comparing their strengths and weaknesses, we provided actionable insights for shop owners, helping them improve stock management, plan for events, and anticipate seasonal trends. Future work could involve gathering more data or refining the models to better handle unexpected events.