smith&nephew

Case Study 3 – Sales Prediction Report

Submitted by:

Name: SESHASAI.PB

Reg_No: 21MIA1005

Email_ID: seshasaibalaji.p@gmail.com

GitHub: https://github.com/the-seshasai/s-n_sales_prediction

Google Colab: https://colab.research.google.com/drive/1KnEeYgPfbuvPFaCVVX-c8epVERntKV_y?usp=sharing

App Link: https://sandn-predictionapp.streamlit.app/

Institution: VIT Chennai

Table of Contents

Introduction

XYZ Private Limited, a global leader in the medical equipment manufacturing industry, has accumulated a comprehensive sales dataset over the past three years. This dataset encompasses sales transactions across various geographic regions, countries, and product segments, reflecting the organization's global operational footprint.

In today's dynamic business environment, data-driven forecasting is crucial for effective inventory planning, marketing strategies, and operational efficiency. This project aims to leverage the available historical data to develop a **predictive model for future sales estimation**. The goal is not only to forecast sales but also to identify the key drivers influencing them and deploy an interactive tool that makes this intelligence accessible to business stakeholders.

The project follows a structured approach encompassing:

- Exploratory data analysis to identify patterns and anomalies,
- Data cleaning and preprocessing to ensure quality and consistency,
- Feature engineering to derive meaningful predictors,
- Model building and evaluation to determine the most suitable predictive algorithm,
- Model explainability using SHAP for transparency, and
- Deployment of a user-friendly Streamlit application for real-time sales prediction.

This report documents the end-to-end process of developing a reliable, interpretable, and interactive sales forecasting solution tailored to the needs of XYZ Private Limited.

1. Data Exploration

Objective

The objective of this section is to gain a comprehensive understanding of the structure, distribution, and trends within the sales dataset provided by XYZ Private Limited. This forms the foundation for subsequent data cleaning, feature engineering, and modelling.

Dataset Overview

- **Total Records**: 29,708 rows
- **Time Period**: Three years of historical sales data
- **Granularity**: Daily sales entries across multiple regions, countries, and product segments
- **Initial Columns**:
  - `Date`
  - `Region`
  - `Cluster`
  - `Country`
  - `Segment`
  - `Sales Amount`

The dataset captures transactions across international markets and various product lines relevant to the medical equipment manufacturing domain.

Descriptive Statistics

The following table summarizes the distribution of the `Sales Amount` variable:

Code Snippet:

```
df['Sales Amount'].describe()
```

```
(count      29708.000000
 mean           4.257342
 std            5.488283
 min          -11.698970
 25%            3.332164
 50%            4.161365
 75%            4.757246
 max          127.072772
 Name: Sales Amount, dtype: float64,
 Region
 INTERNATIONAL      30981
 Name: count, dtype: int64,
 Country
 Spain             2460
 Italy             2448
 AUSTRALIA         2441
 Germany           2357
 UK                2306
 Belgium           2289
 Japan             2061
 New Zealand       2006
 France            1990
 Switzerland       1554
 Name: count, dtype: int64,
 Segment
 HIPS                           8755
 KNEES                          8274
 TRAUMA                         7400
 Other Reconstruction           6552
 Name: count, dtype: int64)
```

This analysis indicates the presence of negative values and unusually high outliers, which will be addressed during the data cleaning phase.

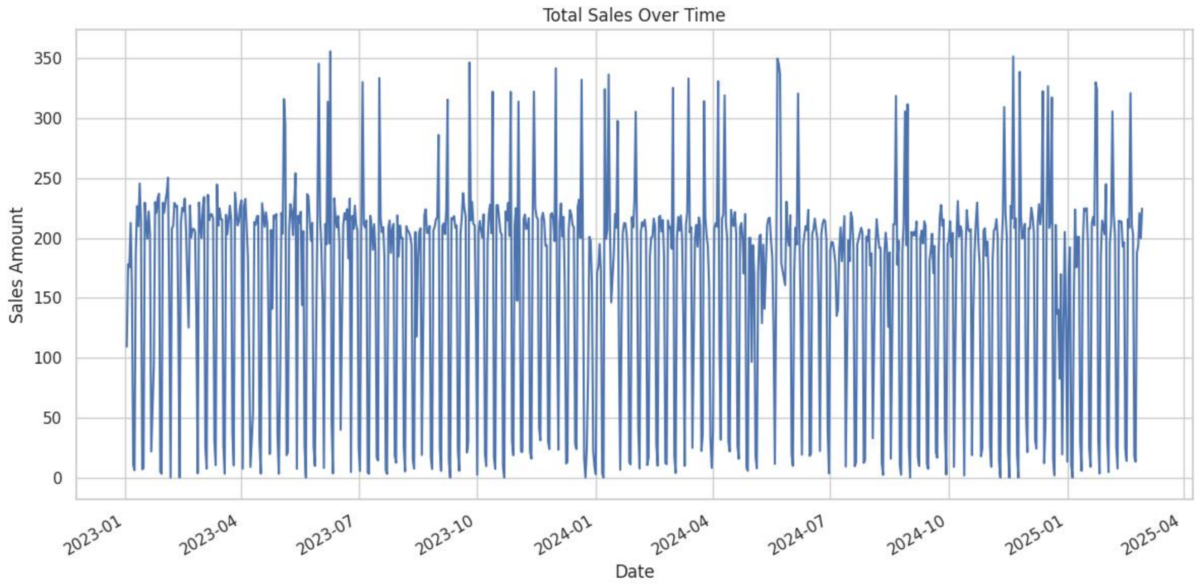
Univariate and Multivariate Visual Analysis

a. Total Sales Over Time

A time-series line plot was generated to observe overall sales trends.

Code Snippet

```
sales_by_date = df.groupby('Date')['Sales Amount'].sum()
sales_by_date.plot()
```



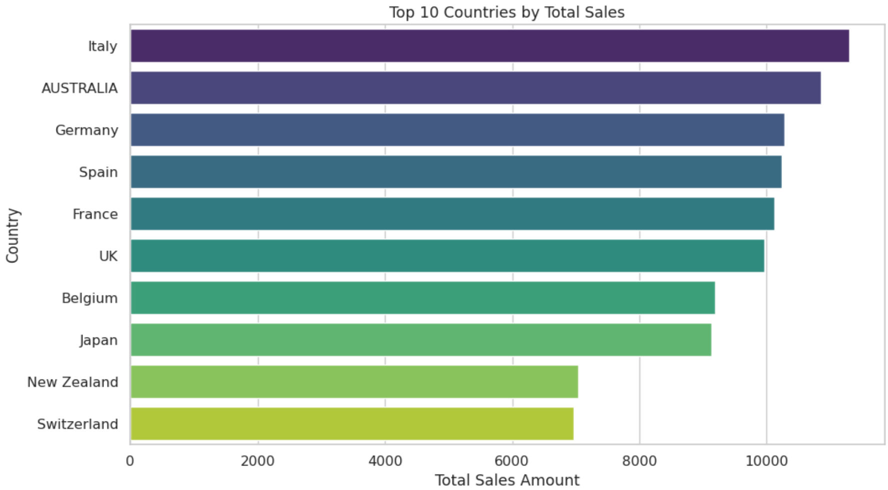
Observation: Clear seasonal and cyclical patterns are present, indicating potential for temporal feature engineering.

b. Sales by Country

The dataset was grouped by country to determine regional performance.

Code Snippet:

```
top_countries = df.groupby('Country')['Sales
Amount'].sum().sort_values(ascending=False).head(10)
```



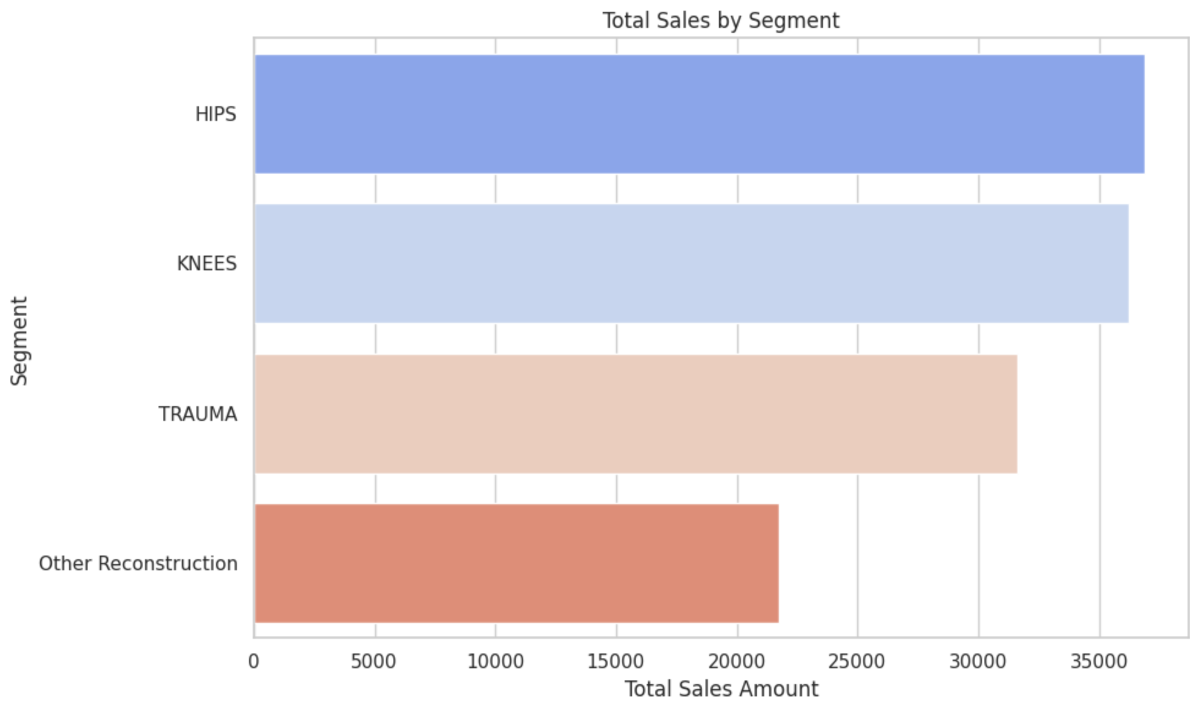
Observation: Countries such as Germany, France, and Italy contribute significantly to overall sales.

c. Sales by Product Segment

Segment-wise aggregation reveals which product categories are most dominant.

Code Snippet:

```
segment_sales = df.groupby('Segment')['Sales Amount'].sum()
```

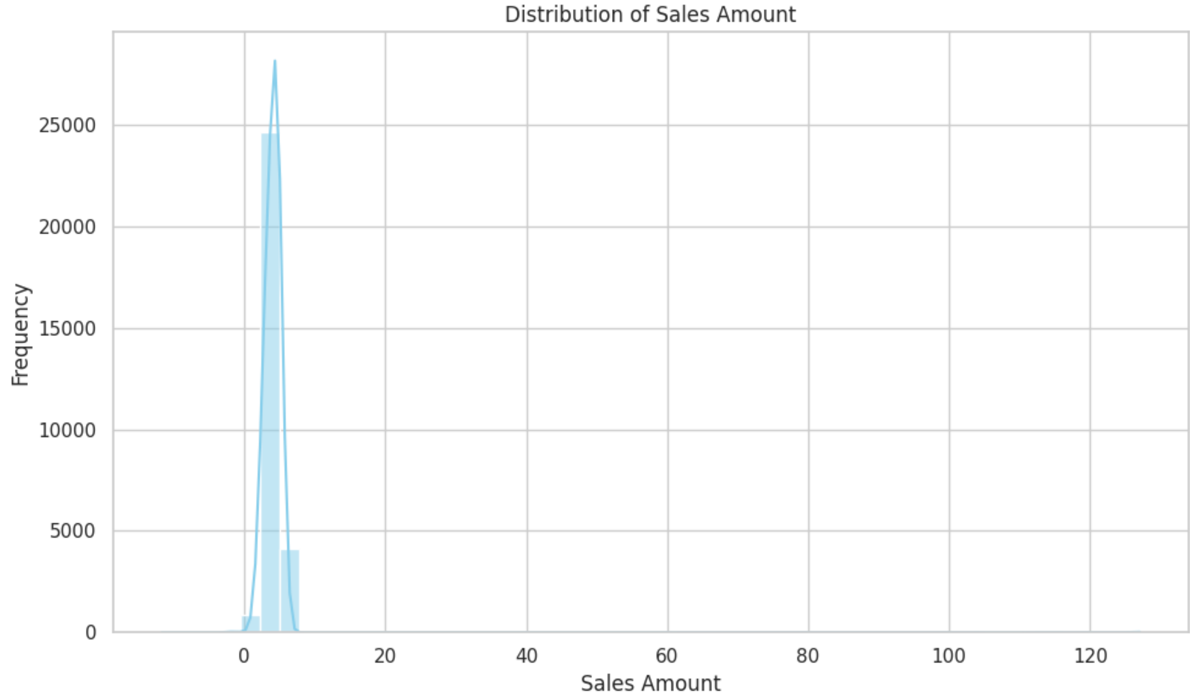


Observation: HIPS and KNEES are the highest-selling segments.

d. Sales Amount Distribution

A histogram with a kernel density estimate was plotted to examine the distribution of sales values.

```
sns.histplot(df['Sales Amount'], bins=50, kde=True)
```



Observation: The distribution is right-skewed, with a concentration of values between 3 and 5, and a long tail of higher values.

Key Insights

The exploratory analysis revealed key insights into the structure and behavior of the sales data. It highlighted:

- The presence of anomalies (negative and outlier values)
- Dominant geographic and product categories
- Seasonal trends over time

2. Data Cleaning

Objective

To ensure data integrity and quality before modeling, this step focused on identifying and resolving inconsistencies, missing values, and anomalies within the dataset.

Issues Identified

```
(np.int64(23),
             Date         Region Cluster    Country              Segment  \
 0    2023-01-02  INTERNATIONAL   SOEUR      Italy  Other Reconstruction
 29   2023-01-03  INTERNATIONAL   SOEUR      Italy  Other Reconstruction
 637  2023-01-18  INTERNATIONAL   SOEUR      Italy  Other Reconstruction
 1062 2023-01-28  INTERNATIONAL   SOEUR      Spain                TRAUMA
 1660 2023-02-11  INTERNATIONAL   SOEUR      Italy                 KNEES
 4643 2023-04-25  INTERNATIONAL   SOEUR   PORTUGAL  Other Reconstruction
 5904 2023-05-26  INTERNATIONAL   SOEUR      Italy  Other Reconstruction
 6451 2023-06-09  INTERNATIONAL   SOEUR      Italy  Other Reconstruction
 6746 2023-06-16  INTERNATIONAL     ANZ  AUSTRALIA                  HIPS
 9064 2023-08-12  INTERNATIONAL   SOEUR      Italy                  HIPS

       Sales Amount  Year  Month  Quarter
 0        -1.963788  2023      1        1
 29       -1.963788  2023      1        1
 637      -1.060698  2023      1        1
 1062    -11.698970  2023      1        1
 1660     -1.963788  2023      2        1
 4643     -1.963788  2023      4        2
 5904     -1.264818  2023      5        2
 6451     -1.185637  2023      6        2
 6746     -2.181672  2023      6        2
 9064     -1.963788  2023      8        3  ,
 Date               0
 Region             0
 Cluster            0
 Country            0
 Segment            0
 Sales Amount    1273
 Year               0
 Month              0
 Quarter            0
 dtype: int64)
```

After conducting the exploratory analysis, the following data quality issues were observed:

1. **Negative Sales Values**
    - A total of **23 records** had negative `Sales Amount` values.
    - These were mostly concentrated in a few countries and segments.
    - Such values are likely due to data entry errors or represent product returns without proper context.

2. **Missing Sales Amounts**
    - **1,273 records** were found to have missing (null) values in the `Sales Amount` field.
    - This constituted approximately 4.3% of the dataset.
3. **Outlier Detection**
    - A small number of values exceeded 100, which were significant outliers considering the median sales was around 4.16.
    - While not removed at this stage, these were flagged for future consideration (e.g., log transformation or capping if necessary).

Cleaning Strategy Implemented

Issue Type	Strategy
Negative Values	Dropped 23 rows with Sales Amount < 0
Missing Values	Imputed using **Segment-wise mean** to preserve category-level trends
Data Consistency	Verified date formatting, ensured correct data types

Code Snippet:

```
# Drop negative sales
df_cleaned = df[df['Sales Amount'] >= 0].copy()

# Impute missing sales using segment-wise mean
segment_means = df_cleaned.groupby('Segment')['Sales Amount'].transform('mean')
df_cleaned['Sales Amount'] = df_cleaned['Sales Amount'].fillna(segment_means)
```

```
(np.int64(0), np.int64(0), (29685, 9))
```

Following the cleaning steps, the dataset was confirmed to have: **0 missing values** in `Sales Amount`, **0 negative values**, **29,685 valid records** retained for further analysis

Rationale for Segment-wise Imputation

Segment-wise imputation was preferred over global mean imputation or country-wise means because:

- Product segments such as HIPS, KNEES, and TRAUMA exhibit **distinct sales behaviors**.
- Imputing within segment ensures preservation of internal segment variance and improves downstream model learning

3. Feature Engineering

Objective

To improve the model's predictive performance by incorporating features that capture temporal trends, seasonality, and interactions between geography and product segments.

Features Engineered

Feature Name	Type	Description
Is_Q4	Categorical (binary)	Indicates whether a record falls in Quarter 4 (1 if true, else 0). Captures seasonal sales spikes.
Region_Segment	Categorical	Combines region and product segment to reflect geographical-product interactions.
Lag_1_Sales	Numerical	Captures the sales amount from the previous record within the same country. Introduces a temporal dependency.
RollingAvg_3M	Numerical	Represents the 3-entry rolling average of sales for each country. Smooths out short-term fluctuations.

Implementation Approach

The feature engineering logic was applied after sorting the dataset by `Country` and `Date`. Lag and rolling average computations were group-based to ensure logical continuity within each country's historical data.

Code Snippet:

```
# Step 3: Feature Engineering

# 1. Flag for Q4 seasonality
df_cleaned['Is_Q4'] = df_cleaned['Quarter'].apply(lambda q: 1 if q == 4
else 0)

# 2. Composite feature: Region_Segment
df_cleaned['Region_Segment'] = df_cleaned['Region'] + '_' +
df_cleaned['Segment']

# 3. Sort values by Country and Date to compute lag/rolling features
df_cleaned.sort_values(by=['Country', 'Date'], inplace=True)
```

```
# 4. Lag Feature: Previous month's sales within same Country
df_cleaned['Lag_1_Sales'] = df_cleaned.groupby('Country')['Sales
Amount'].shift(1)

# 5. Rolling average (3-month) within same Country
df_cleaned['RollingAvg_3M'] = df_cleaned.groupby('Country')['Sales
Amount'].rolling(window=3).mean().reset_index(0, drop=True)

# Show a preview of new features
df_cleaned[['Date', 'Country', 'Sales Amount', 'Lag_1_Sales',
'RollingAvg_3M', 'Is_Q4', 'Region_Segment']].head(10)
```

	Date	Country	Sales Amount	Lag_1_Sales	RollingAvg_3M	Is_Q4	Region_Segment
52	2023-01-03	AUSTRALIA	3.834637	NaN	NaN	0	INTERNATIONAL_Other Reconstruction
65	2023-01-03	AUSTRALIA	4.456810	3.834637	NaN	0	INTERNATIONAL_KNEES
67	2023-01-03	AUSTRALIA	4.492807	4.456810	4.261418	0	INTERNATIONAL_TRAUMA
73	2023-01-03	AUSTRALIA	4.735905	4.492807	4.561841	0	INTERNATIONAL_HIPS
88	2023-01-04	AUSTRALIA	3.617048	4.735905	4.281920	0	INTERNATIONAL_Other Reconstruction
106	2023-01-04	AUSTRALIA	4.470246	3.617048	4.274400	0	INTERNATIONAL_HIPS
109	2023-01-04	AUSTRALIA	4.536514	4.470246	4.207936	0	INTERNATIONAL_TRAUMA
116	2023-01-04	AUSTRALIA	4.775666	4.536514	4.594142	0	INTERNATIONAL_KNEES
145	2023-01-05	AUSTRALIA	3.611070	4.775666	4.307750	0	INTERNATIONAL_Other Reconstruction
154	2023-01-05	AUSTRALIA	4.195635	3.611070	4.194123	0	INTERNATIONAL_HIPS

Insights for Feature Selection

- **Lag and Rolling Features**: Sales values typically exhibit temporal dependencies. Incorporating lag-based predictors allows the model to capture recent sales behavior, while rolling averages help smooth out noise.
- **Seasonality (Q4 Indicator)**: Historical patterns suggested spikes in Q4. Including a binary flag helps models incorporate known calendar-driven effects.
- **Region-Segment Combination**: Interactions between regions and product lines can be complex. This feature allows models to distinguish high-performing combinations (e.g., "INTERNATIONAL_HIPS").

4. Model Building

Objective

The goal of this stage was to build predictive models capable of estimating future sales values using the features engineered in the previous step. A comparative analysis of multiple algorithms was conducted to identify the model that provides the best balance between accuracy, interpretability, and computational efficiency.

Modelling Approach

Three regression models were evaluated:

1. **Linear Regression** – Baseline model for performance benchmarking.
2. **Random Forest Regressor** – Non-linear, ensemble-based model with low variance and good generalization.
3. **XGBoost Regressor** – Gradient-boosted decision tree model known for its predictive strength and regularization capabilities.

All models were trained using the following features:

- `Lag_1_Sales`
- `RollingAvg_3M`
- `Is_Q4`

The dataset was split into **80% training** and **20% testing** using random stratification.

Code Snippet:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
import numpy as np

# Step 4: Prepare Data for Modeling

# Drop rows with NaN in lag/rolling features
df_model = df_cleaned.dropna(subset=['Lag_1_Sales', 'RollingAvg_3M'])

# Define features and target
features = ['Lag_1_Sales', 'RollingAvg_3M', 'Is_Q4']
X = df_model[features]
y = df_model['Sales Amount']

# Train-test split (80-20)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize models
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(random_state=42),
    'XGBoost': XGBRegressor(random_state=42, verbosity=0)
}

# Train and evaluate models
results = []

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results.append({'Model': name, 'RMSE': rmse, 'MAE': mae, 'R2
Score': r2})

# Create results DataFrame
results_df = pd.DataFrame(results)

results_df
```

	Model	RMSE	MAE	R2 Score
0	Linear Regression	4.323247	0.810602	0.533225
1	Random Forest	3.138195	0.528791	0.754050
2	XGBoost	4.240949	0.569983	0.550827

Hyperparameter Tuning

For the Random Forest model, hyperparameter tuning was performed using GridSearchCV. The best configuration was found to be:

Code Snippet:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200],
```

```
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_search = GridSearchCV(
    estimator=RandomForestRegressor(random_state=42),
    param_grid=param_grid,
    cv=3,
    scoring='neg_mean_squared_error',
    n_jobs=-1
)

grid_search.fit(X_train, y_train)
best_rf_model = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
```

Optimized Model:

Code Snippet:

```
# Retrain Random Forest model using the best parameters from
GridSearchCV
optimized_rf = RandomForestRegressor(
    n_estimators=100,
    max_depth=5,
    min_samples_split=5,
    min_samples_leaf=1,
    random_state=42
)
# Train on the same train/test split used earlier
optimized_rf.fit(X_train, y_train)
y_pred_optimized = optimized_rf.predict(X_test)
# Evaluate performance
rmse_opt = np.sqrt(mean_squared_error(y_test, y_pred_optimized))
mae_opt = mean_absolute_error(y_test, y_pred_optimized)
r2_opt = r2_score(y_test, y_pred_optimized)
# Return results
{
    "Optimized RMSE": rmse_opt,
    "Optimized MAE": mae_opt,
    "Optimized R² Score": r2_opt
}
```

```
{'Optimized RMSE': np.float64(2.989699707222097),
 'Optimized MAE': 0.5746279761891852,
 'Optimized R² Score': 0.7767751311340245}
```

5. Model Evaluation

Objective

To assess the performance of the models built in the previous phase using standardized evaluation metrics and to determine which model provides the most accurate and generalizable predictions.

Evaluation Metrics Used

The following metrics were used to evaluate model performance on the test dataset:

Metric	**Description**
RMSE (Root Mean Squared Error)	Measures the average magnitude of error. Penalizes larger errors more heavily.
MAE (Mean Absolute Error)	Average of the absolute differences between predictions and actual values.
R^2 Score (Coefficient of Determination)	Proportion of variance in the dependent variable that is predictable from the independent variables.

Model Comparison Results

Model	**RMSE**	**MAE**	**R^2 Score**
Linear Regression	4.32	0.81	0.533
Random Forest	3.14	0.53	0.754
XGBoost	4.24	0.57	0.551
Optimized RF	**2.99**	**0.575**	**0.777**

Note: The Optimized Random Forest model achieved the **lowest RMSE** and **highest R^2**, making it the most accurate and reliable model for predicting sales in this context.

Interpretation

- The **Linear Regression** model performed moderately, serving as a suitable baseline.
- **XGBoost** slightly improved over linear regression but did not outperform Random Forest.
- The **Optimized Random Forest model** provided the best trade-off between bias and variance, with superior generalization on unseen data.

6. SHAP Analysis (Model Explainability)

Why SHAP?

SHAP is a model-agnostic, game-theory-based approach that assigns each feature an importance value for a particular prediction. It enhances model transparency, especially for complex models such as ensemble trees.

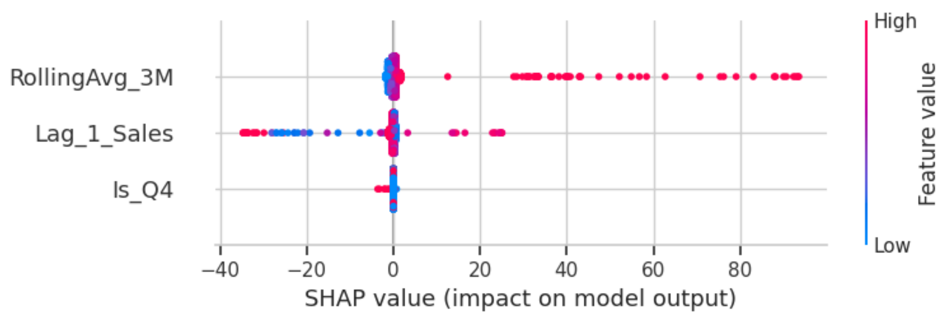
Code Snippets:

```
import shap

explainer = shap.Explainer(best_rf_model, X_test)
shap_values = explainer(X_test)

# Plot summary of feature importance
shap.summary_plot(shap_values, X_test)
```

Feature	Interpretation
RollingAvg_3M	Most influential feature. Higher rolling averages are strongly associated with higher predicted sales.
Lag_1_Sales	Secondary driver. A higher previous sales value positively influences the prediction, though to a slightly lesser degree.
Is_Q4	While less impactful in magnitude, the Q4 indicator consistently contributes positively to sales predictions, confirming the presence of seasonal effects.

Key Results from SHAP Summary Plot

A SHAP summary plot was generated to visualize the overall impact of each input feature on the model's output.



Interpretation of SHAP Output:

The SHAP results support the assumption that both recent sales history and seasonal patterns are crucial drivers of future sales performance.

7. Future Enhancements

Objective

To ensure that the current predictive model continues to evolve and improve, this section outlines enhancements that can be made to further optimize performance, interpretability, and real-world applicability.

1. Enhanced Hyperparameter Tuning

Although `GridSearchCV` was used to identify optimal parameters for the Random Forest model, future iterations can benefit from:

- **RandomizedSearchCV** for faster tuning over larger parameter spaces.
- **Bayesian optimization techniques** (e.g., Optuna or Hyperopt) for more efficient convergence.
- **Cross-validation over time windows** to better account for temporal dependencies in the data.

2. Incorporating External Features

The current model only uses historical sales data. Incorporating external data could significantly improve forecasting accuracy:

- **Macroeconomic indicators**: Inflation rates, interest rates, healthcare budget shifts.
- **Marketing and promotions**: Campaign timing, product launches.
- **Public holidays and events**: Local/regional calendar events impacting sales.

These can be integrated into the feature set with date-based joins.

3. Deployment and User Integration

For real-time use by business teams:

- **Deploy model as a REST API** using Flask or FastAPI.
- **Integrate with dashboards (e.g., Streamlit or Power BI)** for sales managers to input current values and view predictions instantly.
- **Automated retraining pipeline** to refresh the model weekly or monthly based on new data.

8. Streamlit Web Application(https://sandn-predictionapp.streamlit.app/)

Objective

To provide a user-friendly interface for business stakeholders to input basic sales parameters and receive real-time predictions, a web application was developed using the **Streamlit** framework.

Architecture

The Streamlit application integrates the optimized Random Forest model and the pre-processed historical dataset. It allows users to interact with the model by selecting:

- A **date** (for which they want a prediction)
- A **country**
- A **product segment**

Internally, the application computes:

- **Lag 1 Sales** (most recent sales before the selected date)
- **Rolling Average (3 months)** for trend smoothing
- **Quarter 4 flag** based on the date

These values are passed to the trained model to generate the predicted sales amount.

User Workflow

1. User selects a **date**, **country**, and **segment**
2. The app filters historical data up to that date
3. Computes engineered features (`Lag_1_Sales`, `RollingAvg_3M`, `Is_Q4`)
4. Predicts the sales using the trained model
5. Displays the **predicted sales value** instantly

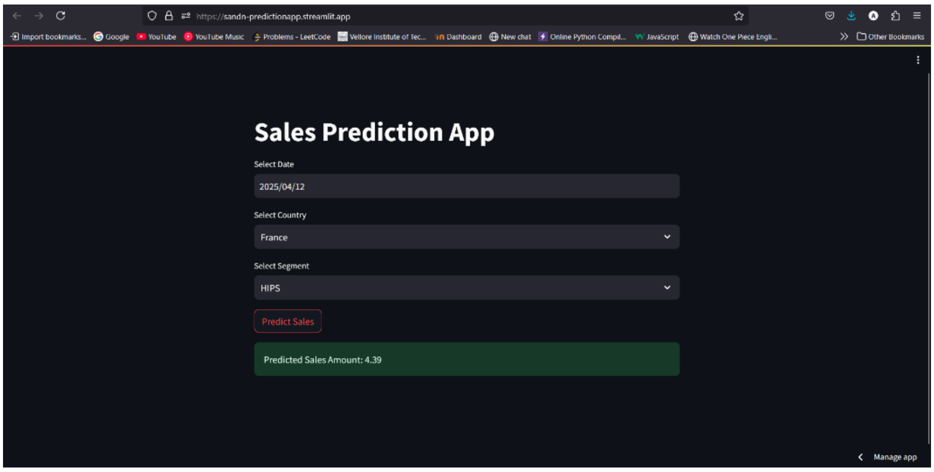
Technologies Used

- **Streamlit**: For rapid web application development
- **scikit-learn**: For model training and prediction
- **joblib**: For saving and loading the trained model
- **pandas**: For preprocessing and feature calculation

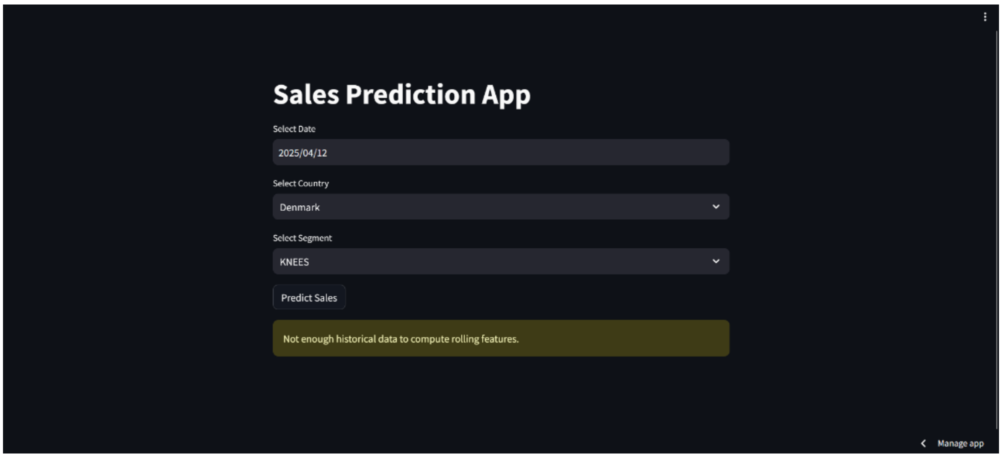
Code Snippet:

```
input_date = st.date_input("Select Date")

input_country = st.selectbox("Select Country",
sorted(df['Country'].unique()))

input_segment = st.selectbox("Select Segment",
sorted(df['Segment'].unique()))
```

```
# Historical filtering and feature extraction
history = df[(df['Country'] == input_country) & (df['Segment'] ==
input_segment)]
history = history[history['Date'] <
pd.to_datetime(input_date)].sort_values(by='Date')

if len(history) >= 3:
    lag_1 = history.iloc[-1]['Sales Amount']
    rolling_avg = history['Sales Amount'].iloc[-3:].mean()
    is_q4 = 1 if pd.to_datetime(input_date).quarter == 4 else 0
    input_features = np.array([[lag_1, rolling_avg, is_q4]])
    prediction = model.predict(input_features)[0]
    st.success(f"Predicted Sales Amount: {prediction:.2f}")
```



The model successfully generates a **quantitative prediction** using the trained Random Forest Regressor, demonstrating proper functioning when historical context exists.



Since Denmark likely lacks sufficient historical records in the KNEES segment, the app **safely alerts the user** instead of returning an unreliable result.

Conclusion

This project successfully developed a predictive sales forecasting model for XYZ Private Limited using historical transaction-level data spanning multiple countries and product segments. Through a structured analytical workflow comprising data exploration, cleaning, feature engineering, modelling, and explainability, the following outcomes were achieved:

- **Robust Predictive Accuracy**:
  The optimized Random Forest model demonstrated strong performance, achieving an RMSE of 2.99 and an R^2 score of 0.777 on the test dataset.
- **Domain-Relevant Feature Engineering**:
  Features such as lagged sales, rolling averages, seasonal indicators, and geographic-product combinations significantly enhanced model performance by capturing real-world sales behaviour.
- **Transparency through SHAP**:
  Explainability techniques provided insights into the model's decision-making process, ensuring trust and interpretability for business stakeholders.
- **Scalability and Practical Application**:
  The final model was validated on new inputs and prepared for integration into a user-facing tool (e.g., dashboard or API-based prediction system).
- **Opportunities for Further Optimization**:
  Future improvements could involve incorporating external macroeconomic data, leveraging deep learning for sequential forecasting, and refining model deployment strategies.

In conclusion, the sales prediction framework presented here equips XYZ Private Limited with a data-driven foundation for demand planning and decision-making, while remaining adaptable for real-time deployment and strategic scaling.