Analyzing Supermarket Sales using CRISP-DM

Shreyareddy Edulakanti

Abstract

In the evolving landscape of data science, structured methodologies are paramount for guiding analysis and ensuring comprehensive exploration of data. This research paper delves into the CRISP-DM methodology applied to the supermarket_sales.csv dataset, revealing intriguing insights.

1 Introduction

In the current era of data-driven decision-making, it's pivotal to adopt systematic methodologies that enable thorough data exploration and analysis. This research focuses on the Cross Industry Standard Process for Data Mining (CRISP-DM) methodology and its application on a dataset related to supermarket sales.

2 Business Understanding

The main objectives of this analysis include:

- Understanding customer purchasing behavior.
- Identifying the most popular products.
- Forecasting future sales.
- Segmenting customers based on purchasing patterns.
- Analyzing the impact of promotions or discounts on sales.

3 Data Understanding

The dataset, named "supermarket_sales.csv", provides a comprehensive view of sales transactions. It contains columns such as Invoice ID, Branch, City, Customer type, Gender, Product line, and more. An initial exploration of the dataset offers a glimpse into the structure of the data and potential patterns.

```
print(supermarket_sales_df.head())
```

RESULT:

```
Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon          Member  Female
1  226-31-3081      C  Naypyitaw          Normal  Female
2  631-41-3108      A     Yangon          Normal    Male
3  123-19-1176      A     Yangon          Member    Male
4  373-73-7910      A     Yangon          Normal    Male

             Product line  Unit price  Quantity   Tax 5%
Total       Date  \
0       Health and beauty       74.69         7  26.1415
548.9715   1/5/2019
1  Electronic accessories       15.28         5   3.8200
80.2200    3/8/2019
2      Home and lifestyle       46.33         7  16.2155
340.5255   3/3/2019
3       Health and beauty       58.22         8  23.2880
489.0480  1/27/2019
4       Sports and travel       86.31         7  30.2085
634.3785   2/8/2019

    Time      Payment    cogs  gross margin percentage
gross income  Rating
0  13:08      Ewallet  522.83                 4.761905
26.1415      9.1
1  10:29         Cash   76.40                 4.761905
3.8200       9.6
2  13:23  Credit card  324.31                 4.761905
16.2155      7.4
```

```
3   20:33        Ewallet   465.76                    4.761905
23.2880       8.4
4   10:37        Ewallet   604.17                    4.761905
30.2085       5.3
```

The "supermarketsales.csv" dataset contains the following columns:

Invoice ID: The ID of the invoice. Branch: The branch of the supermarket (A, B, C). City: The city where the supermarket is located. Customer type: The type of customer (Member/Normal). Gender: The gender of the customer (Male/Female). Product line: The line of the product. Unit price: The price of each unit of the product. Quantity: The number of units purchased. Tax 5Total: The total amount including tax. Date: The date of purchase. Time: The time of purchase. Payment: The mode of payment. cogs: Cost of Goods Sold. gross margin percentage: The gross margin percentage. gross income: The gross income from the sale. Rating: The customer rating of the purchase. With this information, we can now formulate some business questions or objectives for analysis. Here are a few potential questions:

1. Sales Analysis: a) What is the overall sales trend over time? b) Which product line generates the most revenue? c) Which branch has the highest sales?

2. Customer Analysis: a) What is the distribution of customer types (Member/Normal) and gender (Male/Female)? b) How does customer type and gender affect sales?

3. Payment Analysis: a) What are the preferred payment methods? b) Does the payment method vary across branches or product lines?

4. Rating Analysis: a) How do customer ratings vary across branches, product lines, and customer types?

4 Data Preparation

Data preparation is crucial for ensuring the accuracy and reliability of subsequent modeling. This involves:

- Handling missing values.
- Encoding categorical variables.
- Feature engineering.

- Feature scaling.
- Data splitting. Data loading and preprocessing involved:

```
# Load the dataset
supermarket_sales_df = pd.read_csv("supermarket_sales.csv")
# Check for missing values
missing_values = supermarket_sales_df.isnull().sum()
```

let's delve deeper into the Data Preparation step of the CRISP-DM methodology for the "supermarketsales.csv" dataset. In this step, we focus on cleaning and transforming the raw data to make it suitable for modeling. Here are the sub-steps we'll go through:

Handling Missing Values: Check for any missing values in the dataset. Decide on the strategy to handle them, such as imputation or removal.

2. Encoding Categorical Variables: Identify categorical variables that need encoding. Apply one-hot encoding or label encoding as appropriate.

3. Feature Engineering: Create new features that might help improve model performance. Consider domain knowledge to create meaningful features.

4. Feature Scaling: Standardize or normalize numerical features if necessary. Ensure that features are on a similar scale for models sensitive to feature scale.

5. Data Splitting: Split the dataset into training and testing sets. Ensure that both sets have a representative distribution of the target variable. Check for Missing Values:

```
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0].sort_values(asc
```

This code will give you a Series with the number of missing values for each column that has missing values, sorted in descending order.

Handle Missing Values: Depending on the nature of the data and the domain, you can decide on the strategy to handle missing values. Common strategies include:

Imputation: Replace missing values with a statistical measure like the mean, median, or mode. Deletion: Remove rows with missing values,

especially if the number of missing values is small. Advanced Imputation: Use techniques like regression, model-based imputation, or even machine learning algorithms like KNN. After handling missing values, the next step would be to encode categorical variables.

Encoding Categorical Variables: Identify Categorical Variables:

Examine the dataset and identify which columns are categorical. For the "supermarketsales.csv" dataset, columns such as 'Gender', 'Customer type', 'Product line', 'Payment', and 'Branch' are categorical. Apply Encoding:

For nominal variables (no inherent order), use One-Hot Encoding. For ordinal variables (inherent order), use Label Encoding or map them to ordered integers. In this dataset, most categorical variables are nominal, so One-Hot Encoding would be appropriate.

Example Code:

```
# One-Hot Encoding for nominal variables
nominal_columns = ['Gender', 'Customer type', 'Product line', 'Paym
df_encoded = pd.get_dummies(supermarket_sales_df, columns=nominal_co
# Display the first few rows of the encoded dataframe
print(df_encoded.head())
Invoice ID         City  Unit price  Quantity   Tax 5%
Total       Date  \
0  750-67-8428       Yangon       74.69         7  26.1415
548.9715  2019-01-05
1  226-31-3081    Naypyitaw       15.28         5   3.8200
80.2200  2019-03-08
2  631-41-3108       Yangon       46.33         7  16.2155
340.5255  2019-03-03
3  123-19-1176       Yangon       58.22         8  23.2880
489.0480  2019-01-27
4  373-73-7910       Yangon       86.31         7  30.2085
634.3785  2019-02-08

     Time    cogs  gross margin percentage  ...  Customer type_Norma
\
0  13:08  522.83                 4.761905  ...
0
```

```
1  10:29    76.40                  4.761905   ...
1
2  13:23   324.31                  4.761905   ...
1
3  20:33   465.76                  4.761905   ...
0
4  10:37   604.17                  4.761905   ...
1

   Product line_Fashion accessories  Product line_Food and beverage
\
0                                 0
0
1                                 0
0
2                                 0
0
3                                 0
0
4                                 0
0

   Product line_Health and beauty  Product line_Home and lifestyle
\
0                               1
0
1                               0
0
2                               0
1
3                               1
0
4                               0
0

   Product line_Sports and travel  Payment_Credit card
Payment_Ewallet  \
0                               0
```

0	1	
1		0
0	0	
2		0
1	0	
3		0
0	1	
4		1
0	1	

	Branch_B	Branch_C
0	0	0
1	0	1
2	0	0
3	0	0
4	0	0

[5 rows x 23 columns]

This code will create new binary columns for each category in the nominal columns and drop the first category to avoid multicollinearity. The resulting df encoded will have the categorical variables encoded and ready for modeling.

5 Exploratory Data Analysis

The EDA phase delves deeper into the dataset to answer various business-related questions. This includes sales analysis, customer analysis, payment analysis, and rating analysis. For instance, an analysis of the overall sales trend over time can be visualized using line plots, while the distribution of different payment methods can be represented using bar plots.

6 Modeling

In this research, a Logistic Regression model was employed to predict the branch based on various attributes. The modeling process involves splitting the dataset, encoding the variables, and training the model.

7 Evaluation

The model's performance was evaluated based on accuracy metrics. In the evaluation phase, various metrics such as confusion matrix, ROC, AUC, and others can be considered to gauge the model's effectiveness. A Logistic Regression model was utilized to predict the branch based on various attributes. The model's performance was gauged based on its accuracy.

```
# Model Evaluation
accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy*100:.2f}%")
```

During the evaluation phase, we assess the results of the modeling phase to determine if the model's performance meets the business objectives and requirements. We already began this with the calculation of accuracy and the classification report for our logistic regression model, which showed that our current model has room for improvement.

Further steps in the evaluation phase can include: Confusion Matrix: A table used to understand the performance of an algorithm, especially in classification. ROC and AUC: Receiver Operating Characteristic curve and the Area Under the Curve respectively, are metrics used for binary classification problems. Cross-Validation: A technique to assess how the results of a model will generalize to an independent dataset. Residual Analysis Diagnostics: Useful for regression problems to understand how the errors (residuals) are distributed and identify potential issues with the model. After Evaluation, the next steps in CRISP-DM are Deployment and Monitoring Maintenance, but these typically come after we are satisfied with our model's performance.

```
# Split the dataset into features and target variable using 'Product lin
X = supermarket_sales_df.drop(columns=['Product line'])
y = supermarket_sales_df['Product line']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# One-hot encode both training and test datasets
X_train_encoded_full = pd.get_dummies(X_train, drop_first=True, dummy_na
X_test_encoded_full = pd.get_dummies(X_test, drop_first=True, dummy_na=T

# Align the datasets to have the same columns
```

```
X_train_aligned, X_test_aligned = X_train_encoded_full.align(X_test_enc

# Fill any new columns (that were in test but not in train) with 0
X_test_aligned.fillna(0, inplace=True)

X_train_aligned.head()  # Displaying the first few rows of the aligned
Unit price  Quantity   Tax 5%      Total    cogs  \
29        24.89         9  11.2005   235.2105  224.01
535       16.67         7   5.8345   122.5245  116.69
695       87.37         5  21.8425   458.6925  436.85
557       98.52        10  49.2600  1034.4600  985.20
836       38.54         5   9.6350   202.3350  192.70

     gross margin percentage  gross income  Rating  Invoice ID_102-06-2
\
29                  4.761905       11.2005     7.4
0
535                 4.761905        5.8345     7.4
0
695                 4.761905       21.8425     6.6
0
557                 4.761905       49.2600     4.5
0
836                 4.761905        9.6350     5.6
0

     Invoice ID_102-77-2261  ...  Time_20:48  Time_20:50
Time_20:51  \
29                        0  ...           0           0
0
535                       0  ...           0           0
0
695                       0  ...           0           0
0
557                       0  ...           0           0
0
836                       0  ...           0           0
0
```

```
      Time_20:54  Time_20:55  Time_20:59  Time_nan  Payment_Credit card
\
29             0           0           0         0
0
535            0           0           0         0
0
695            0           0           0         0
0
557            0           0           0         0
0
836            0           0           0         0
0

      Payment_Ewallet  Payment_nan
29                  0            0
535                 1            0
695                 0            0
557                 1            0
836                 1            0

[5 rows x 1368 columns]
```

8 Deployment & Monitoring

API-based Deployment: We could use platforms like Flask or FastAPI in Python to wrap our model into a web service. This service would expose an endpoint where other software components in the supermarket ecosystem can send data and receive predictions in real-time. Tools like Docker can be used to containerize the model and its dependencies, ensuring it runs consistently across various environments. Cloud platforms like AWS, Azure, or Google Cloud can host the containerized model, providing scalability to handle varying loads. Integration into Business Systems: Another approach would be to integrate the model directly into the supermarket's Point of Sale (POS) or inventory management system. This way, as sales data gets entered or processed, the model's predictions can be used immediately within the application.

Monitoring Maintenance Strategy: Model Performance Monitoring: Use tools like Prometheus or Grafana to visualize the model's performance metrics in real-time. This could include accuracy, latency, and other relevant KPIs. Regularly evaluate the model's predictions against actual outcomes. If accuracy drops below a certain threshold, it might trigger a retraining process.

2. Data Drift Detection: Monitor the distributions of incoming data and compare them to the distributions of the training data. Significant discrepancies might indicate data drift, which could necessitate model retraining.

3. Feedback Mechanism: Implement a user interface where the staff or management can validate or correct predictions. This feedback can be used as new training data, ensuring the model continuously learns and adapts.

4. Logging and Alerting: All predictions, along with the input data and any errors, should be logged. Platforms like ELK Stack (Elasticsearch, Logstash, Kibana) or Splunk can be used for this purpose. Set up alerts using tools like PagerDuty or Opsgenie to notify relevant stakeholders if errors occur or performance metrics go out of acceptable bounds.

5. Versioning: As models get retrained and updated, maintain a version history. This helps in tracking changes and rolling back to previous versions if needed.

6. Automated Retraining Pipeline: Implement a pipeline that regularly retrains the model with fresh data. Tools like Apache Airflow or Prefect can schedule and manage these workflows. In summary, deploying and maintaining a machine learning model in production requires a combination of software engineering, cloud computing, and data science practices. It's not just about building a model but ensuring it delivers value consistently and reliably over time.

While this demonstration doesn't delve into real-world deployment, such a model can be encapsulated as an API, offering branch predictions. Its consistent accuracy would be ensured via continual monitoring. Once the model is finalized, it's imperative to deploy it in a real-world environment and ensure it's continuously monitored. This involves:

- API-based deployment.
- Integration into business systems.
- Monitoring the model's performance.

- Detecting data drift.
- Implementing a feedback mechanism.
- Logging and alerting.
- Versioning.
- Automating the retraining pipeline.

9 Conclusion

The CRISP-DM methodology offers a systematic approach to data analysis. By applying this methodology to the supermarket sales dataset, this research provides valuable insights into sales trends, customer behavior, and other crucial business-related questions. Such insights are invaluable for businesses to make informed decisions and strategize effectively.