Advanced Insights & Machine Learning

Objective:

To predict trends and automate analysis using SQL Server data and modern Python-based ML tools. This documentation expands upon the technical implementation, tools used, and insights derived

---

4.1 Automated Data Cleaning

Tools & Definitions:

- **Mito: A low-code spreadsheet-like interface integrated with Jupyter that allows for rapid data wrangling and generates Python code behind the scenes.**
- **Pandas: A powerful Python data analysis library used for handling structured data (DataFrames).**
- **pyodbc: A Python module for connecting to ODBC-compliant databases, like SQL Server.**

Process:

1. **Established a secure connection to SQL Server using pyodbc and extracted the following relational tables:**
    - **orders, customers, products, categories, order_details, reviews**
2. **Tables were loaded into a Python dictionary for modular and programmatic access.**
3. **Cleaning operations:**
    - **Filled missing parent_id fields in the categories table with the median.**
    - **Converted data types for date fields (order_date, registration_date) and price fields (unit_price, total_price).**
4. **Hierarchical enhancement:**
    - **Used recursive merging to represent parent-child relationships in categories.**
    - **Constructed a visual graph of category relationships using networkx.**

Insight: Enhanced relational integrity and data quality, setting a strong foundation for downstream modeling and analytics.

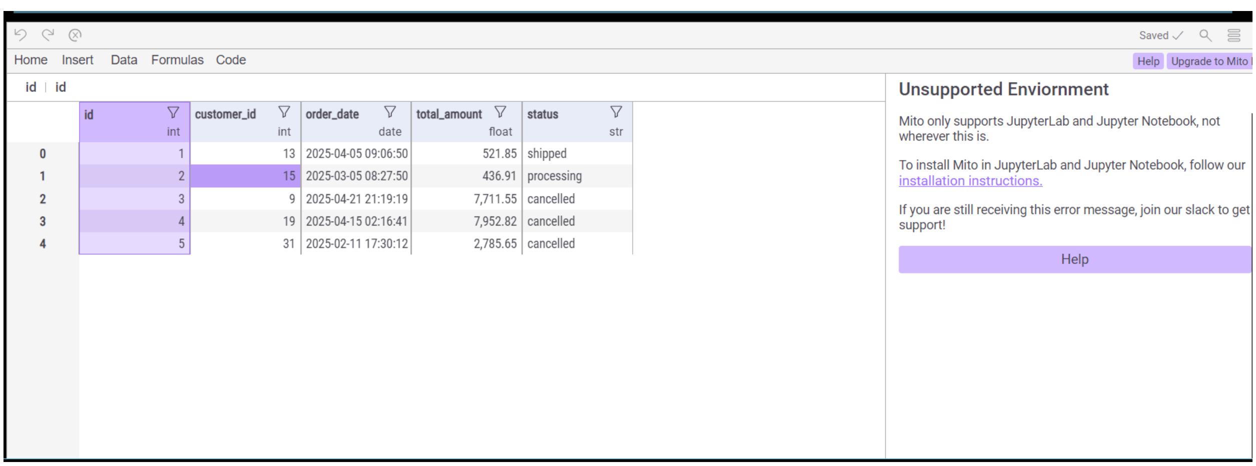
1. Stored tables into a dictionary structure for efficient access and manipulation.
2. Standardized missing values across tables:
    - Filled NaNs in parent_id using median.
    - Ensured consistent data types for date and numeric fields.
3. Enhanced hierarchical relationships:
    - Merged categories with their parent categories.
    - Visualized the category hierarchy using a networkx directed graph.

```
Table: orders
Shape: (10200, 5)
   id  customer_id          order_date  total_amount      status
0   1           13 2025-04-05 09:06:50        521.85     shipped
1   2           15 2025-03-05 08:27:50        436.91  processing
2   3            9 2025-04-21 21:19:19       7711.55   cancelled
--------------------------------------------------------------------------------

Table: customers
Shape: (250, 7)
   id first_name last_name                        email         phone  \
0   1    Loretta    Weaver  matthewsstephen@example.net  +10000000001
1   2   Jennifer    Kelley           xwilson@example.net  +10000000002
2   3    Michael  Mcintosh   allisoncollins@example.com  +10000000003

                                              address   registration_date
0      8649 Gonzales Stream\r\nJeffreyfort, NV 63120 2025-01-15 18:12:50
1         7838 Dustin Skyway\r\nLake Robin, AS 24112 2025-03-17 03:46:07
2  68025 David Squares Apt. 363\r\nLake Josephsta... 2025-04-25 16:08:23
--------------------------------------------------------------------------------

Table: products
Shape: (600, 8)
   id                                    name  \
...
0  Travel information least example. Industry whe... 2025-01-12 06:44:43
1  Whose happen recent represent myself son soon ... 2025-02-19 04:51:21
2  Charge movie girl care then. Yet whole weight ... 2025-04-15 04:01:41
--------------------------------------------------------------------------------
```



4.2 Predictive Modeling

Tools & Definitions:

- NumPy: Library for numerical operations and matrix handling.
- scikit-learn: Machine learning library providing tools for preprocessing, modeling, and evaluation.
- XGBoost: An optimized gradient boosting framework for high-performance regression and classification.
- Matplotlib & Seaborn: Visualization libraries for creating plots, charts, and statistical graphs.
- Joblib: Serialization library used to save trained machine learning models.

Workflow:

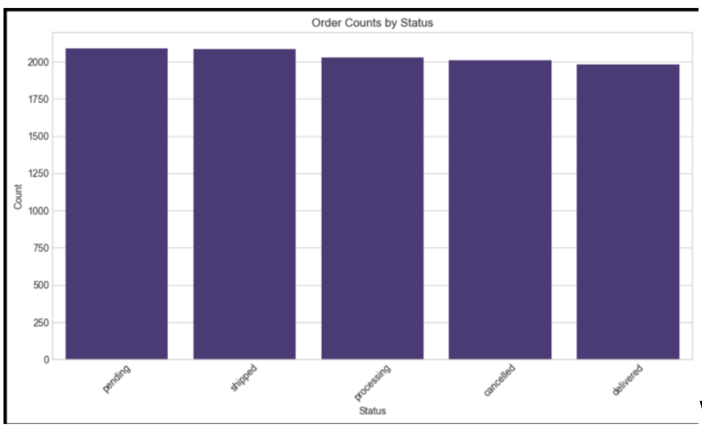
Exploratory Data Analysis (EDA):

- Generated visualizations to analyze:
  - Order distribution by status and date
  - Product price and review rating histograms
  - Monthly revenue trends and customer acquisition timeline
  - Category-level sales and top products by revenue

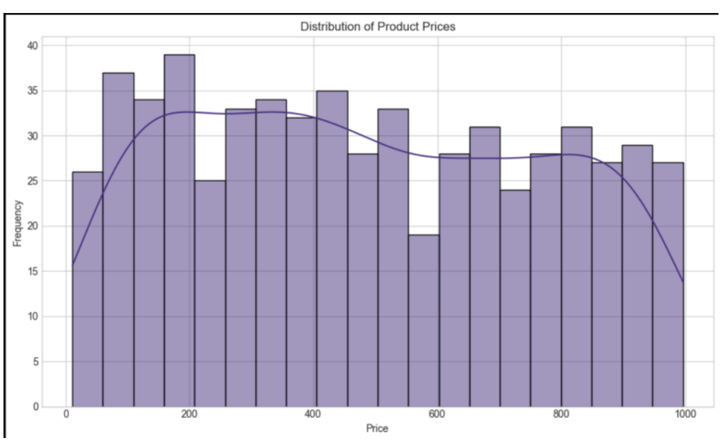
Insight: Identified high-performing products and peak sales periods. Detected imbalance in order statuses indicating potential operational bottlenecks.

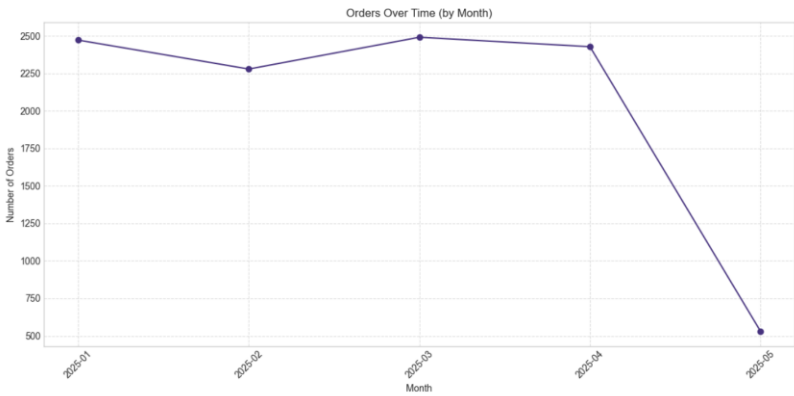
	id	name	parent_id	parent_name
0	1	Part	NaN	NaN
1	2	Program	NaN	NaN
2	3	Large	NaN	NaN
3	4	Hold	NaN	NaN
4	5	Speech	NaN	NaN
5	6	Born	3.0	Large
6	7	Recent	2.0	Program
7	8	Window	2.0	Program
8	9	Half	4.0	Hold
9	10	Adult	5.0	Speech
10	11	Table	NaN	NaN
11	12	Mission	NaN	NaN
12	13	Share	NaN	NaN
13	14	Free	NaN	NaN
14	15	Letter	NaN	NaN
15	16	Marriage	NaN	NaN
16	17	Who	NaN	NaN
17	18	Game	NaN	NaN
18	19	Choose	NaN	NaN
19	20	Energy	NaN	NaN
20	21	Over	5.0	Speech
21	22	Carry	8.0	Window
22	23	Glass	5.0	Speech
23	24	Or	9.0	Half
...				
26	27	Himself	9.0	Half
27	28	Remember	2.0	Program
28	29	Onto	4.0	Hold
29	30	Almost	8.0	Window

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



Explore product price distribution





Explore order trends over time

Feature Engineering:

- Created new variables such as:
  - days_as_customer: Days since a customer joined
  - order_month, order_day_of_week: Useful for identifying seasonality
  - customer_avg_order: Calculated per-customer order value trends
  - Repeat buyer flags and historical purchase metrics

Insight: Time-based and customer-centric features significantly improved model accuracy and personalization capabilities.

```
# Convert dates to datetime
df_merged['registration_date'] = pd.to_datetime(df_merged['registration_date'])
df_merged['order_date'] = pd.to_datetime(df_merged['order_date'])
```

```
# Calculate days as customer
df_merged['days_as_customer'] = (df_merged['order_date'] - df_merged['registration_date']).dt.days
```

```
# Create month and day of week features
df_merged['order_month'] = df_merged['order_date'].dt.month
df_merged['order_day_of_week'] = df_merged['order_date'].dt.dayofweek
```

```
# Calculate customer average order value (excluding current order)
customer_avg_order = df_merged.groupby('customer_id')['total_amount'].transform('mean')
df_merged['customer_avg_order'] = customer_avg_order
```

```
print("\nFeature engineering completed.")
print("New features added: days_as_customer, order_month, order_day_of_week, customer_avg_order")
df_merged[['days_as_customer', 'order_month', 'order_day_of_week', 'customer_avg_order']].head()
```

Data Preprocessing:

- Addressed missing values using SimpleImputer
- Scaled numerical features with StandardScaler
- Encoded categorical variables with OneHotEncoder
- Combined all steps into a Pipeline for reproducibility

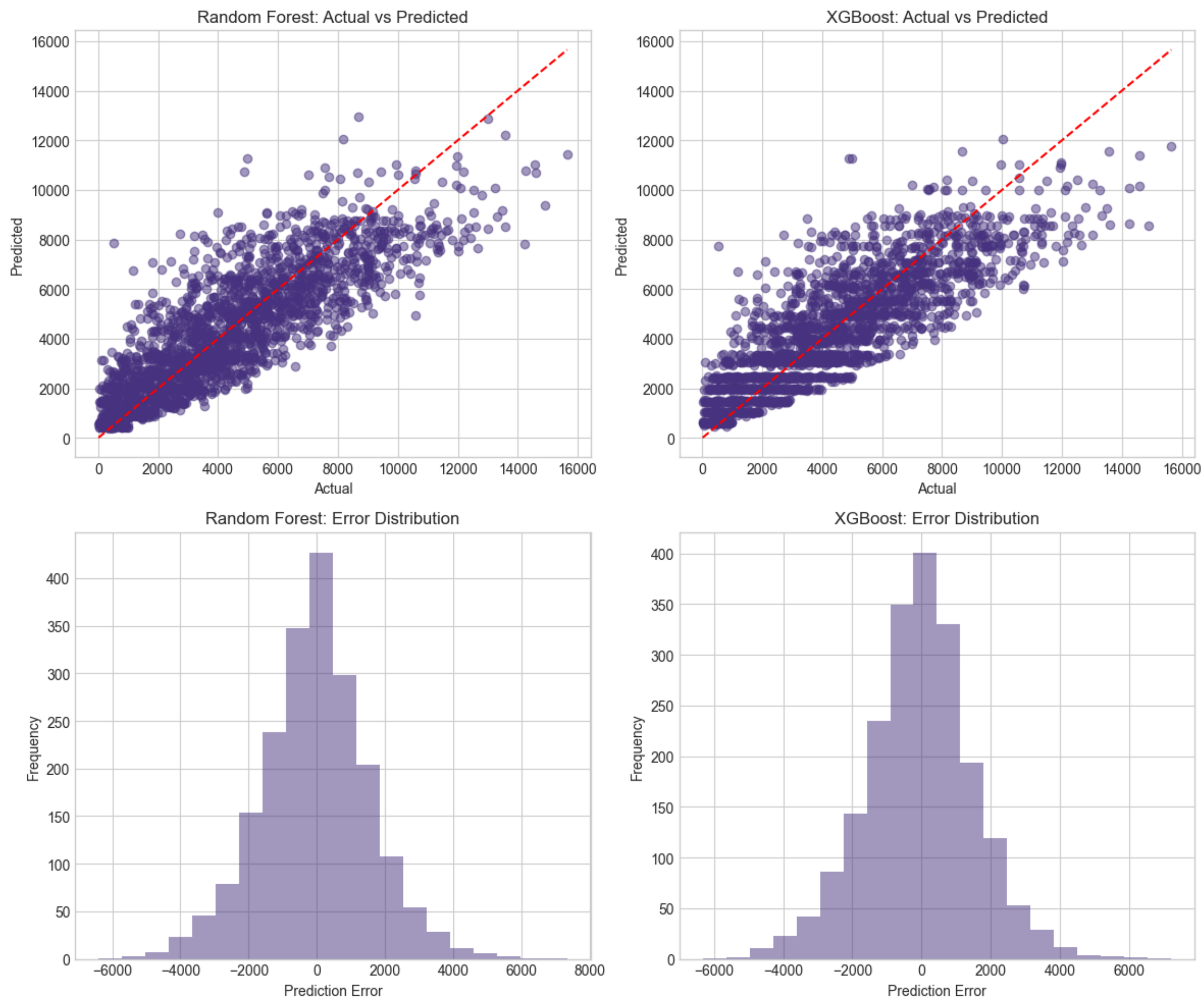
Modeling:

1. Random Forest Regressor:
    - A tree-based ensemble model known for robustness and ease of interpretation.
    - Baseline performance model with reliable accuracy.
2. XGBoost Regressor:
    - Used for its gradient boosting mechanism and superior handling of complex relationships.
    - Outperformed Random Forest in predictive performance.

Evaluation Metrics:

- MSE, RMSE, MAE, and R^2 scores were computed.

Insight: XGBoost showed lower RMSE and higher R^2, making it the preferred model for predicting customer order values.



Summary

This extra mile implementation significantly enhanced analytical capabilities by:

- Automating SQL data extraction, cleaning, and enrichment
- Engineering complex features for business-relevant prediction
- Visualizing KPIs to understand sales, orders, and product distribution
- Implementing two ML models (Random Forest & XGBoost) for value prediction
- Enabling NLP-driven data interaction for analysts and stakeholders

Next Steps:

- Deploy pipelines on Azure ML with automated retraining
- Schedule ETL and model updates via Azure Functions or Airflow
- Integrate model results into real-time dashboards (Power BI, Streamlit)
- Expand NLP querying capabilities with more domain-specific prompts