**Mage-AI machine learning model development**

**Time Series Analysis**

**Introduction**

Time series analysis is a statistical technique used to analyze patterns and behaviors within sequential data points ordered over time. In essence, it focuses on understanding how data evolves and changes over time. This branch of analysis is crucial for extracting meaningful insights, making forecasts, and understanding underlying trends and patterns present in time-dependent data.

Time series data can be collected at various frequencies: hourly, daily, weekly, monthly, yearly, etc.

**Importance of Time Series Analysis**

Time series analysis holds significant importance across a multitude of disciplines due to its ability to uncover hidden patterns, forecast future trends, and make data-driven decisions. Some key reasons for its importance include:

1. **Forecasting**: Time series analysis enables forecasting future values based on historical data, helping organizations make informed decisions and plan effectively.
2. **Pattern Recognition**: By analyzing past trends and patterns, time series analysis helps identify recurring behaviors, cycles, and anomalies within the data.
3. **Risk Management**: In finance and economics, time series analysis is crucial for risk assessment, portfolio management, and predicting market fluctuations.
4. **Operations Management**: In industries such as manufacturing, retail, and logistics, time series analysis aids in demand forecasting, inventory optimization, and resource allocation.
5. **Healthcare**: In healthcare, time series analysis is used for patient monitoring, disease outbreak prediction, and medical resource planning.
6. **Environmental Science**: Time series analysis helps in studying climate patterns, pollution levels, and natural disaster prediction.

**Applications in Various Fields**

Time series analysis finds applications in diverse domains, including but not limited to:

* **Finance**: Stock market analysis, asset pricing, risk management.
* **Economics**: Macroeconomic indicators, GDP forecasting, inflation analysis.
* **Marketing**: Sales forecasting, customer behavior analysis, campaign effectiveness.
* **Signal Processing**: Speech recognition, image processing, sensor data analysis.
* **Meteorology**: Weather forecasting, climate modeling, extreme event prediction.
* **Healthcare**: Disease surveillance, patient monitoring, healthcare resource allocation.
* **Engineering**: Quality control, fault detection, predictive maintenance.

The versatility and applicability of time series analysis make it an indispensable tool for extracting insights from temporal data in almost every industry and scientific discipline.

By leveraging time series analysis techniques, practitioners can gain valuable insights, make accurate predictions, and drive informed decision-making processes in their respective fields.

**Definition of Time Series Data**

Time series data consists of a collection of data points, where each data point corresponds to a specific time period. This could be regular intervals (e.g., hourly, daily, monthly) or irregular intervals depending on the context of the data collection. Time series data is often represented as a series of pairs (time, value), where time represents the timestamp or period when the observation was made, and value represents the measurement or observation recorded at that time.

**Demo project: Using Merge AI**

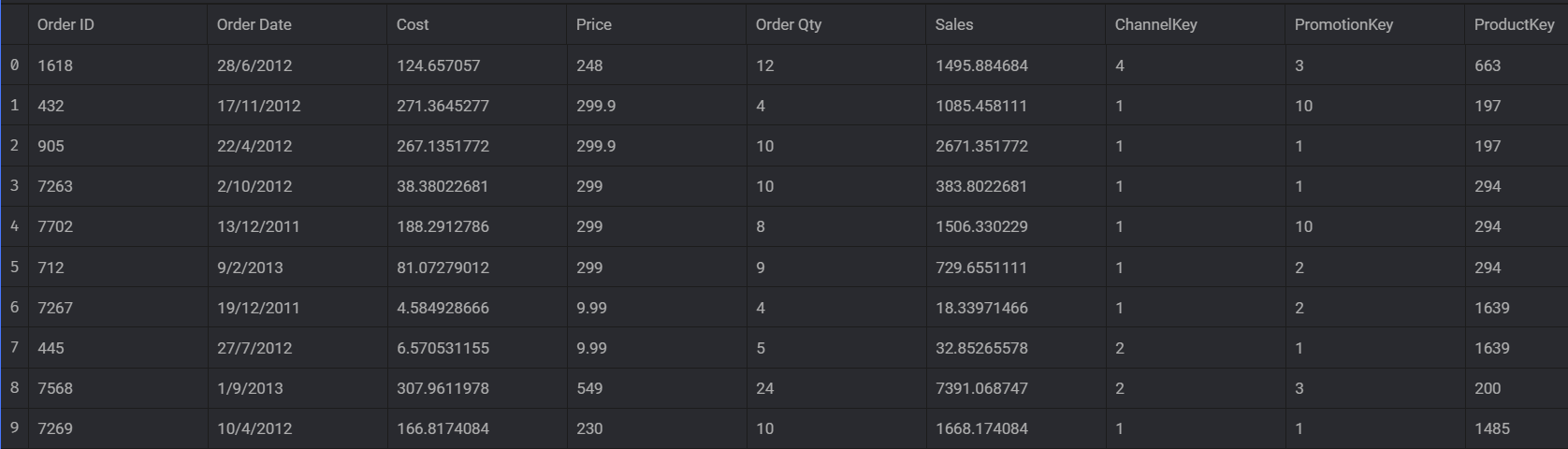
Project summary

This project focuses on analyzing sales data to provide valuable insights for presentation to management and investors. The primary objectives were to clean and preprocess the data, develop sales forecasting models, and ensure the generalization of these models. In here use mage.ai platform to do those.

**Dataset Overview**

Data set - [Sales\_Data1.xlsx - Google Sheets](https://docs.google.com/spreadsheets/d/1hH-5QVDV2iljBiTySlAPW6kONFHZuS8j/edit?pli=1#gid=867219703)

* Sales\_data



* Channel\_data

A black rectangular object with a black line

Description automatically generated with medium confidence

* Product\_data

A screenshot of a computer

Description automatically generated

* Category\_data

A screenshot of a computer

Description automatically generated

Top of Form

**Data Cleaning & Preprocessing:**

Data preprocessing is a crucial step in building machine learning models. It involves

transforming and cleaning raw data to make it suitable for model training. Proper data

preprocessing can significantly impact the performance and accuracy of your models.

Steps Taken:

1. Loading Datasets:

* Load sales, channel, product, and category data

1. Merge data:

* Merged datasets using common identifiers (ChannelKey, ProductKey, ProductSubCategoryKey) to create a unified dataset for analysis.

1. Handline Duplicates: use mage.ai inbuild transformer

* Duplicates can arise due to various reasons such as data collection errors, system malfunctions, or repeated measurements. It's essential to identify and appropriately handle duplicates to avoid biasing analysis results and obtaining inaccurate insights.

1. Handling Missing Values: use mage.ai inbuild transformer

* Identified and addressed missing values in each dataset.

1. Outlier handling: use mage.ai inbuild transformer
2. Remove unwanted Columns: use mage.ai inbuild transformer

* Remove Columns-'ChannelKey','PromotionKey','ProductKey','ProductSubCategoryKey','StateID','Product Sub Category','ProductName', 'Manufacturer'

1. Clean column names: use mage.ai inbuild transformer

**Clean Column Names**

Cleans column names according to the following rules:

1. Names are converted to snake case
2. All wrapping whitespace or underscores are removed
3. All non-alphanumeric characters (except ”\_”) are removed
4. Numbers are prefixed with “number\_”
5. All Python keywords are postfixed by ”\_”

Snake case is chosen as

1. all characters are lower case
2. no whitespace is used (which makes referring to columns in code easier)
3. Check the data types of all the columns in the dataset : use mage.ai custom block

A computer screen shot of a code

Description automatically generated

1. Convert The ‘order\_date’ column :

which is originally in a timestamp format with both date and time to a string and truncate the time information.

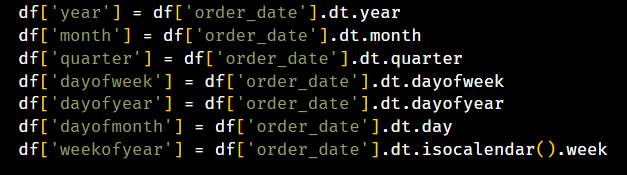


1. Label encoder:

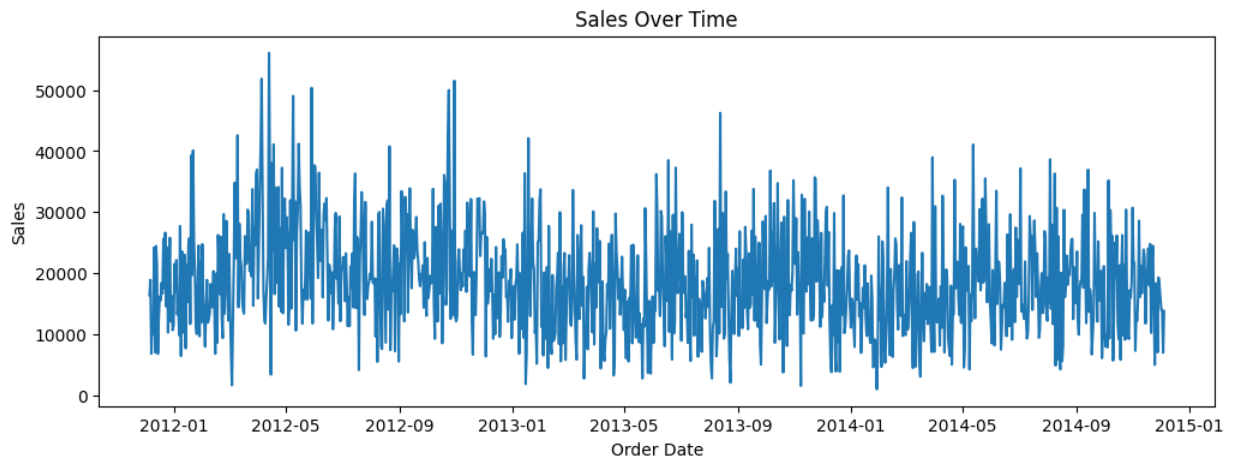
* Label encoding is a technique used in machine learning to convert categorical variables into numerical format. In time series analysis, label encoding can be particularly useful when dealing with categorical features or target variables that are not inherently numeric.

1. Feature Engineering:

* Created new features based on sales date and time to provide additional insights.

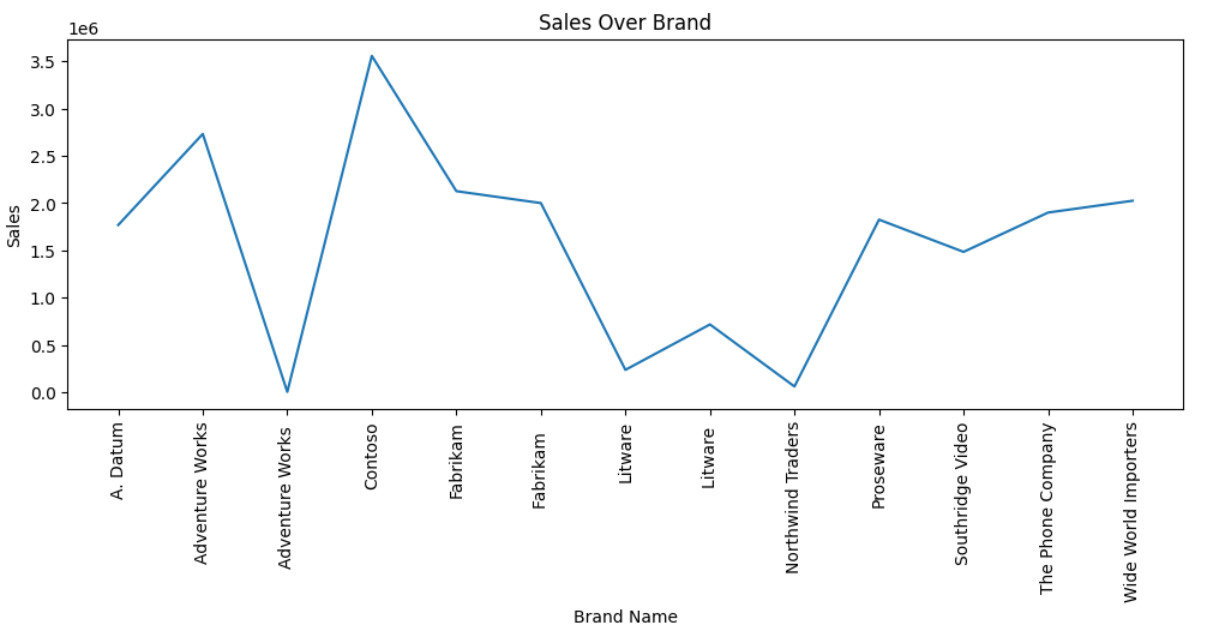


**Visualizations**



A graph with a line

Description automatically generated



A graph with blue lines

Description automatically generated

**Model Development**

* Split data

split\_date = '2014-06-01'

*# Split the data into training and testing sets*

    train = data[data['order\_date'] < split\_date]

    test = data[data['order\_date'] >= split\_date]

* Features and Target

*Features* – year, month , quarter, dayofweek, dayofyear, dayofmonth, weekofyear, CategoryName, BrandName, ChannekName

*Target* – sales

* Use a XGBoost regressor with specified hyperparameters.

reg = xgb.XGBRegressor(

        base\_score=0.5,

        booster='gbtree',

        n\_estimators=10000,

        early\_stopping\_rounds=50,

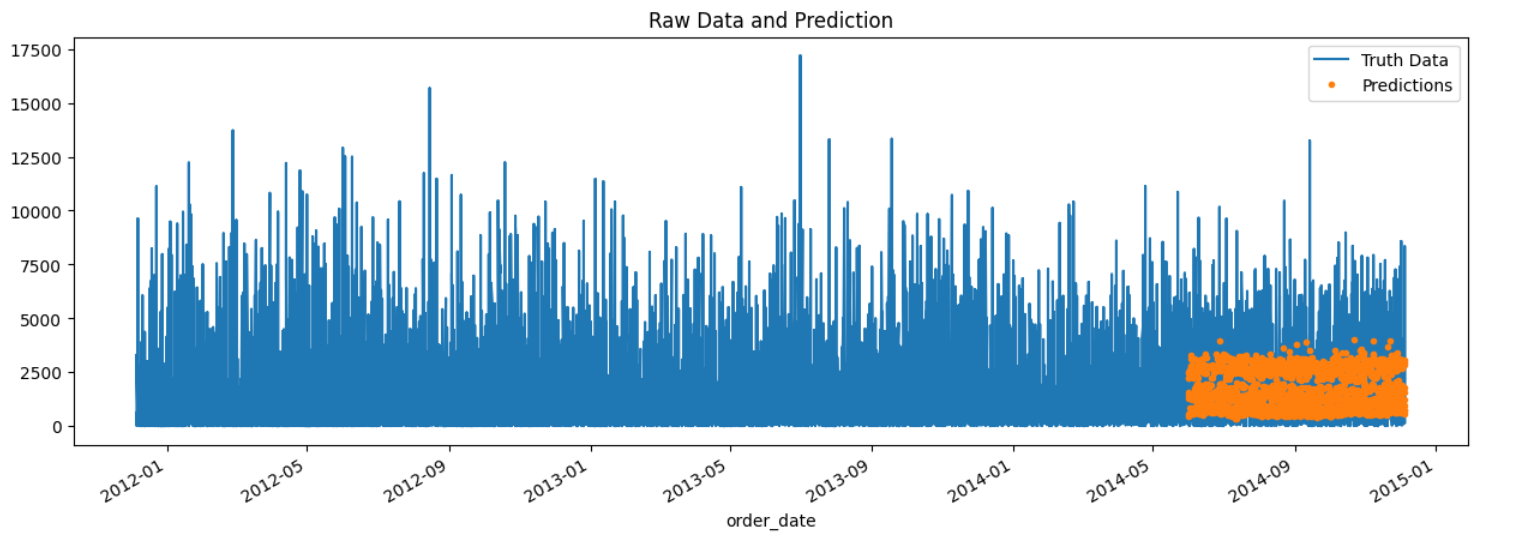
        objective='reg:linear',

        max\_depth=3,

        learning\_rate=0.01

    )

* Fit the model.
* Calculate evaluation metrics for training set.
* Save the model as pickle file.
* Export and save the testing dataset.



**XGBOOST**

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm that belongs to the family of gradient boosting techniques. It is widely used for classification and regression tasks due to its speed, accuracy, and scalability. XGBoost can be effectively used for time series analysis tasks such as forecasting and anomaly detection. It can be adapted for time series analysis by engineering suitable features and using appropriate evaluation metrics.

**1. Feature Engineering:**

* **Lag Features**: Create lag features by shifting the target variable or other relevant features backward in time to capture temporal dependencies.
* **Rolling Statistics**: Compute rolling statistics such as moving averages, rolling standard deviations, or exponential moving averages to capture trends and patterns over time.
* **Time-Related Features**: Extract time-related features such as day of the week, month, season, or time of day to account for seasonality and periodic patterns.

**2. Train-Test Split:**

Split the time series data into training and testing sets, ensuring that the training data contains historical observations while the testing data contains future observations that need to be predicted.

**3. Model Training:**

* **Define Features and Target Variable**: Prepare the feature matrix (X) and target variable (y) from the training data.
* **Initialize XGBoost Model**: Initialize an XGBoost regression model (**xgb.XGBRegressor()**) or classification model (**xgb.XGBClassifier()**) depending on the nature of the prediction task.
* **Train the Model**: Fit the XGBoost model to the training data using the **fit()** method, specifying appropriate hyperparameters such as the learning rate, maximum depth of trees, and number of estimators.

**4. Model Evaluation:**

* **Forecasting Evaluation Metrics**: Use appropriate evaluation metrics for time series forecasting tasks, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or Symmetric Mean Absolute Percentage Error (SMAPE).
* **Anomaly Detection**: For anomaly detection tasks, evaluate the model's performance using metrics such as precision, recall, F1-score, or area under the ROC curve (AUC-ROC).

**5. Model Deployment:**

Deploy the trained XGBoost model to make predictions on unseen data, such as future time points in the time series. Monitor the model's performance over time and retrain/update the model periodically as needed.

***References***

[XGBoost Documentation — xgboost 2.0.3 documentation](https://xgboost.readthedocs.io/en/stable/)

[How to predict a time series using XGBoost in Python (setscholars.net)](https://setscholars.net/how-to-predict-a-time-series-using-xgboost-in-python/#:~:text=In%20summary%2C%20setting%20up%20an%20XGBoost%20model%20for,and%20making%20predictions%20with%20new%20time%20series%20data.)

[How to Use XGBoost for Time Series Forecasting - MachineLearningMastery.com](https://machinelearningmastery.com/xgboost-for-time-series-forecasting/)