Product sales forecasting

Problem Statement

In the competitive retail industry, the ability to predict future sales accurately is crucial for operational and strategic planning. Product sales forecasting aims to estimate the number of products a store will sell in the future, based on various influencing factors such as store type, location, regional characteristics, promotional activities, and temporal variations (such as holidays and seasons). This project focuses on developing a predictive model that uses historical sales data from different stores to forecast sales for upcoming periods.

Need and Use of Product Sales Forecasting

Effective sales forecasting is fundamental for multiple aspects of retail management and operation, including:

1. Inventory Management: Accurate sales forecasts help ensure that stores maintain optimal inventory levels—enough to meet customer demand without overstocking, which can lead to increased costs or waste, especially in the case of perishable goods.
2. Financial Planning: Forecasting sales allows businesses to estimate future revenue and manage budgets more effectively. This is crucial for allocating resources to areas such as marketing, staffing, and capital investments.
3. Marketing and Promotions: Understanding when sales peaks and troughs are likely to occur enables retailers to plan effective marketing campaigns and promotional offers to boost revenue or manage customer flow.
4. Supply Chain Optimization: Sales forecasts inform production schedules, logistics, and distribution plans, ensuring that products are available where and when they are needed, thereby reducing transportation and storage costs.
5. Strategic Decision Making: Long-term sales forecasting supports broader business strategies, including store expansions, market entry, and other capital expenditures.

Dataset: https://drive.google.com/drive/folders/1fBQ1PlWMho3kHF9qXrD0McZNfpJIcbrn

Data description

1. ID: Unique identifier for each record in the dataset.
2. Store_id: Unique identifier for each store.
3. Store_Type: Categorization of the store based on its type.
4. Location_Type: Classification of the store's location (e.g., urban, suburban).
5. Region_Code: Code representing the geographical region where the store is located.
6. Date: The specific date on which the data was recorded.
7. Holiday: Indicator of whether the date was a holiday (1: Yes, 0: No).
8. Discount: Indicates whether a discount was offered on the given date (Yes/No).

9. #Order: The number of orders received by the store on the specified day.
10. Sales: Total sales amount for the store on the given day.

Block 1: Tableau Visulisation

Data visualization plays a crucial role in uncovering trends, patterns, and insights within large datasets, particularly in retail sales where multiple factors influence outcomes. Tableau offers powerful tools for creating interactive and dynamic visualizations that can help stakeholders understand complex data and make informed decisions. For the Product Sales Forecasting project, utilizing Tableau will enable the visualization of sales data across various dimensions such as time, location, and store characteristics.

Suggestions for Tableau Visualizations and Dashboards

1. Sales Performance Dashboard:
    - Time Series Analysis: Line charts showing daily, weekly, or monthly sales trends. Include the ability to drill down from year to month to day.
    - Comparison by Store Type and Location: Use bar charts or pie charts to display sales distribution by store type and location type, helping to identify which categories are performing better.
2. Regional Sales Analysis:
    - Region-Specific Performance: Provide detailed charts that compare regions based on sales volume, number of orders, and average order size.
3. Promotional Impact Analysis:
    - Discount Effectiveness: Scatter plots or box plots comparing sales on days with discounts versus non-discount days.
    - Holiday Sales Impact: Bar charts showing sales performance on holidays compared to regular days.
4. Operational Insights Dashboard:
    - Daily Orders versus Sales: Scatter plots to analyze the relationship between the number of orders and sales, potentially highlighting efficiency or operational issues.
    - Stock Management Insights: Indicators that may suggest overstocking or stock shortages based on rapid changes in daily sales figures.
5. Interactive Forecast Evaluation:
    - Forecast vs. Actual Sales: Line charts showing both forecasted and actual sales to evaluate the accuracy of the forecasting models.
    - Error Metrics Visualization: Graphs displaying forecasting error metrics such as MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) over time to assess model performance.
6. Customizable Filters:

- Dynamic Filtering Options: Allow users to filter data on the dashboard by date range, store type, region code, and whether the day was a holiday or had a discount, facilitating user-driven exploration of the data.

Block 2: EDA and Hypothesis testing

Exploratory Data Analysis (EDA) and hypothesis testing are foundational steps in understanding the dynamics and factors that influence sales in retail. EDA provides a systematic approach to uncovering trends, detecting outliers, and visualizing relationships within the dataset, while hypothesis testing allows for testing assumptions about these relationships statistically. These processes are vital in building a robust model for forecasting product sales.

Suggestions for EDA

1. Univariate Analysis: Study the distribution of each variable, such as daily sales and the number of orders, using histograms or box plots to understand their central tendencies and dispersions.
2. Bivariate Analysis: Examine relationships between sales and potentially influential factors like discounts, holidays, and store types using scatter plots and correlation matrices.
3. Time Series Analysis: Analyze sales trends over time to identify seasonality, trends, and cyclic behavior.
4. Categorical Data Analysis: Use bar charts and frequency tables to explore the distribution of sales across different store types, locations, and regions.
5. Handling Missing Values: Identify any missing data in the dataset and decide on appropriate methods for imputation based on the nature of the data.
6. Outlier Detection: Detect and handle outliers in the dataset since they can significantly skew the results of the data analysis and model predictions.

Suggestions for Hypothesis Testing

1. Impact of Discounts on Sales:
    - Hypothesis: Stores offering discounts will have significantly higher sales than stores not offering discounts.
    - Test: Perform a t-test for the mean sales on days with discounts versus days without.
2. Effect of Holidays on Sales:
    - Hypothesis: Sales on holidays are higher compared to non-holidays.
    - Test: Use a t-test or ANOVA to compare sales on holidays vs. regular days.
3. Sales Differences Across Store Types:
    - Hypothesis: Different store types experience different sales volumes.
    - Test: Conduct ANOVA to compare the mean sales across different store types.

4. Regional Sales Variability:
    - Hypothesis: There is significant variability in sales across different regions.
    - Test: Kruskal-Wallis test if the data is not normally distributed or ANOVA if it is, to compare sales across regions.
5. Correlation between Number of Orders and Sales:
    - Hypothesis: A higher number of orders correlates with higher sales.
    - Test: Calculate Pearson or Spearman correlation coefficient, depending on the data distribution.

Block 3 ML Modeling

Machine Learning (ML) modeling in the context of product sales forecasting involves developing predictive algorithms that can accurately forecast future sales based on historical data and various influencing factors. By harnessing ML techniques, retailers can gain valuable insights into future sales trends, enabling proactive business decisions to optimize inventory, staffing, and marketing strategies.

Steps for ML Modeling in Product Sales Forecasting

1. Data Processing

- Data Cleaning: Address missing values, remove duplicates, and correct inconsistencies in the dataset.
- Feature Engineering: Develop new features that could enhance the model's predictive power, such as time-based aggregates (e.g., sales in the last week), ratios, or interaction terms between features.
- Data Transformation: Scale numerical features and encode categorical variables to prepare the data for modeling. Techniques like normalization or standardization and one-hot encoding or label encoding can be applied.
- Train-Test Split: Divide the data into training and testing sets to ensure the model can be objectively evaluated.

2. Model Selection

- Baseline Model: Start with a simple model to establish a baseline performance. Linear regression is a common choice.
- Complex Models: Explore more sophisticated models to improve accuracy. Potential models include:
    - Time Series Models: ARIMA, SARIMA, or Prophet, which are specifically designed for forecasting based on time series data.
    - Tree-Based Models: Random forests and gradient boosting machines (e.g., XGBoost, LightGBM) that can handle nonlinear relationships and interactions between features.

    - Deep Learning Models: Neural networks, especially LSTM (Long Short-Term Memory) models, which are well-suited for sequences like time series data.
- Ensemble Techniques: Combine predictions from multiple models to improve accuracy and robustness.

3. Model Evaluation and Validation

- Cross-Validation: Implement time-series specific cross-validation techniques to evaluate model performance over different temporal splits of the data.
- Performance Metrics: Use metrics appropriate for regression tasks, such as MAE (Mean Absolute Error), MSE (Mean Squared Error), RMSE (Root Mean Squared Error), and MAPE (Mean Absolute Percentage Error).
- Residual Analysis: Analyze the residuals to ensure there are no patterns left unmodeled.

Block 4: Deployment

Purpose of Deployment via Flask API

- Operational Integration: Seamlessly integrate the forecasting model into business operations, allowing for real-time predictions that can inform decision-making processes.
- Accessibility: Make the predictive capabilities accessible to various frontend interfaces, such as business dashboards, mobile apps, or other enterprise software systems.
- Showcasing your project to recruiters

Steps for Deploying a Machine Learning Model via Flask API

1. Preparation and Setup

- Environment Setup: Create a Python virtual environment and install Flask along with necessary libraries such as numpy, pandas, scikit-learn, and any other libraries used in the model.
- Directory Structure: Organize your Flask application with proper directories for templates, static files, and the main application script.

2. Model Serialization

- Save the Model: Train your model and save the serialized version using pickle or joblib. This file will be loaded by the Flask application to make predictions.
- Load Model Function: Implement a function in your Flask app to load this serialized model into memory when the server starts. This avoids reloading the model with each request, enhancing performance.

3. API Development

- API Endpoints: Define routes in your Flask application that handle requests. Typically, you'll need at least one route for receiving input data and returning predictions.
- Request Handling: Set up your endpoint to accept data (e.g., JSON format), which includes the features required by the model to make predictions.
- Response: Ensure that the API properly formats the response, returning the predicted sales figures along with any relevant confidence intervals or error metrics.

Note:

- The suggestions/Ideas provided above are intended to assist you. The primary aim is to offer guidance on what aspects can be analyzed. If no valuable insights can be derived from a particular analysis, feel free to skip it.