Superstore Sales Prediction Project Documentation

Project Overview

The Superstore Sales Prediction project is a comprehensive machine learning initiative designed to forecast sales for a retail superstore using historical transaction data. The objective is to deliver actionable insights for optimizing inventory management, pricing strategies, and revenue forecasting. The project encompasses data preprocessing, feature engineering, model development, retraining, and deployment of a Flask-based web application for real-time sales predictions, enhanced with MLOps integration and a dashboard for business insights.

Data Description

The dataset, sourced from a Superstore, contains 9,994 transactions with 21 columns, providing detailed transactional information. Below is a structured overview of the dataset:

Column Name	Description	Data Type	Example Values
Row ID	Unique transaction identifier	Integer	1, 2, 3, ...
Order ID	Unique order identifier	String	CA-2013-152156
Order Date	Date of order placement	Date (Excel)	41587 (Jan 6, 2013)
Ship Date	Date of order shipment	Date (Excel)	41590 (Jan 9, 2013)
Ship Mode	Shipment method	String	Second Class, Standard Class
Customer ID	Unique customer identifier	String	CG-12520
Customer Name	Customer name	String	Claire Gute
Segment	Customer segment	String	Consumer, Corporate
Country	Transaction country	String	United States
City	Transaction city	String	Henderson
State	Transaction state	String	Kentucky
Postal Code	Transaction postal code	Integer	42420
Region	Geographical region	String	South, West
Product ID	Unique product identifier	String	FUR-BO-10001798
Category	Product category	String	Furniture, Technology
Sub-Category	Product subcategory	String	Bookcases, Chairs
Product Name	Product name	String	Bush Somerset Collection Bookcase
Sales	Total sales amount	Float	261.96
Quantity	Units sold	Integer	2
Discount	Discount percentage applied	Float	0.0
Profit	Transaction profit	Float	41.9136

Key Data Insights

- **Temporal Scope**: Multi-year transaction data enables time-series analysis.
- **Geographical Scope**: All transactions are from the United States, segmented by region.
- **Financial Metrics**: Includes Sales, Quantity, Discount, and Profit for financial analysis.
- **Product Diversity**: Covers three categories (Furniture, Office Supplies, Technology) with multiple subcategories.
- **Data Integrity**: Contains anomalies (e.g., negative profits, sales > 10,500 removed in retraining) requiring preprocessing.

Methodology

Libraries and Tools

The project leverages a comprehensive set of Python libraries:

- **Core**: NumPy, Pandas
- **Visualization**: Matplotlib, Seaborn, Yellowbrick
- **Machine Learning**: Scikit-learn, XGBoost, LightGBM, CatBoost, NGBoost
- **Preprocessing**: Feature-engine, Scikit-learn’s ColumnTransformer, PowerTransformer, SimpleImputer
- **MLOps**: MLflow for experiment tracking and model logging
- **Web Framework**: Flask for web application deployment
- **Utilities**: Joblib, Warnings, Logging, OS

Data Preprocessing

Feature Engineering

- **Derived Features**: Extracted `Order Day`, `Order Month`, `Order Weekday`, and `Ship Day` from `Order Date` and `Ship Date` using Pandas datetime operations.
- **Removed Features**: Dropped `Discount Percentage` and `Operating Expenses` to align with features available at prediction time.
- **Final Features**: `Profit`, `Postal Code`, `Discount`, `Order Day`, `Order Month`, `Quantity`, `Ship Day`, `Order Weekday`.

Data Cleaning

- **Outlier Removal**: Removed transactions with `Sales > 10,500` to address extreme outliers, as implemented in `retrain.py`.
- **Missing Values**: Handled missing values using `SimpleImputer` with a median strategy for columns with nulls in both training and test sets.

Transformations

- **Log Transformation**: Applied to `Quantity` and `Profit` using `FunctionTransformer(np.log1p)` to stabilize variance.
- **Square Root Transformation**: Applied to `Discount` using `FunctionTransformer(np.sqrt)` to reduce skewness.
- **Remainder Features**: Passed through unchanged using `ColumnTransformer`'s `remainder='passthrough'` option.

Scaling

- Applied `StandardScaler` to normalize all features, ensuring consistent scale for model training.

Model Development

- **Pipeline**: Constructed a Scikit-learn `Pipeline` in `retrain.py` with:
  - `ColumnTransformer` for feature-specific transformations.
  - `StandardScaler` for feature normalization.
  - `ExtraTreesRegressor` as the final model.
- **Model Selection**: `ExtraTreesRegressor` was selected after evaluating multiple regression models (see Model Comparison) for its superior performance.
- **Hyperparameter Tuning**:
  - Used `RandomizedSearchCV` with a reduced parameter grid for efficiency:
    - `n_estimators`: [200, 500]
    - `criterion`: ['squared_error', 'absolute_error']
    - `max_features`: ['auto', 'sqrt']
    - `bootstrap`: [True]
    - `oob_score`: [True, False]
    - `max_samples`: [0.75, 1]
  - Configured with 3-fold cross-validation, 4 iterations, and parallelized execution (`n_jobs=-1`).
  - Achieved an R^2 score of 97.6% post-tuning.
- **Cross-Validation**: Performed 5-fold cross-validation on the test set, logging the mean R^2 score to MLflow.
- **Model Serialization**: Saved the pipeline as `pipeline.pkl` in the `models` directory and as `pipeline.pkl` in the root directory for MLflow compatibility. The pipeline was logged to MLflow using `mlflow.sklearn.log_model`.

Web Application

- **Framework**: Built using Flask, as implemented in `app.py`, with templates for user interaction.
- **Functionality**:
  - **Home Page**: Allows users to input features (e.g., `Profit`, `Postal Code`, `Order Date`) for real-time sales predictions.
  - **Prediction Route**: Processes input data, applies the pipeline's transformations, and returns predictions using the trained `ExtraTreesRegressor`.
  - **Dashboard**: Provides business insights with KPIs (Sales, Profit, Quantity, Profit Margin) and visualizations for categories, products, regions, and shipping modes, sourced from `Superstore.xlsx`.
  - **MLOps Page**: Displays MLflow experiment history and model performance logs.
  - **API Endpoint**: `/api/dashboard_data` serves JSON data for dynamic dashboard updates, filtered by customer segment.
- **MLOps Integration**: Uses MLflow to log prediction inputs, outputs, and errors during inference, with run history retrieved via `MlflowClient`.
- **Logging**: Model performance and errors are logged to `logs/model_retraining.log` and `logs/model_performance_info.log`.

Model Comparison

A comprehensive evaluation of various regression models was conducted to identify the best performer. The models were assessed using key metrics: R^2 Score, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). The table below summarizes the performance, with `ExtraTreesRegressor` outperforming others.

Model	R^2 Score	MSE	MAE	MAPE	Strengths	Weaknesses
ExtraTreesRegressor	97.6 %	Low	Low	Low	Highly accurate, handles non-linear relationships, robust to outliers	Computationally intensive, less interpretable
RandomForestRegressor	~95%	Medium	Medium	Medium	Robust, handles non-linear data, reduces overfitting	Slower than ExtraTrees, sensitive to tuning
GradientBoostingRegressor	~94%	Medium	Medium	Medium	Strong predictive power, handles complex relationships	Sensitive to overfitting, requires tuning
XGBoostRegressor	~94%	Medium	Medium	Medium	Fast, scalable, handles missing data well	Complex tuning, can overfit

LightGBMRegressor	~93%	Medium	Medium	Medium	Fast, efficient for large datasets, handles categorical data	Requires careful tuning, less interpretable
CatBoostRegressor	~93%	Medium	Medium	Medium	Handles categorical features natively, robust to overfitting	Slower training compared to LightGBM
LinearRegression	~80%	High	High	High	Simple, interpretable, fast training	Poor performance on non-linear data
Lasso	~80%	High	High	High	Feature selection, prevents overfitting	Limited to linear relationships
Ridge	~80%	High	High	High	Handles multicollinearity, stable	Limited to linear relationships
KNeighborsRegressor	~85%	Medium	Medium	Medium	Non-parametric, simple to implement	Sensitive to feature scaling, computationally expensive
SVR	~82%	High	High	High	Effective for non-linear data with kernels	Slow on large datasets, sensitive to scaling

Conclusion

The Superstore Sales Prediction project delivers a robust, high-performing machine learning solution for sales forecasting. By integrating advanced preprocessing, feature engineering, a comprehensive model comparison, an automated retraining pipeline, and a feature-rich Flask web application with MLOps and business intelligence capabilities, the project provides a strong foundation for retail analytics with significant potential for future enhancements.