# Sales Forecasting via Time Series Regression (Trend & Seasonality)


## Abstract

This report addresses the development and application of time series regression methods—specifically baseline linear and polynomial regression with seasonal features—to predict store sales using historical sales data. The work includes exploratory data analysis (EDA), feature engineering, model training, and performance evaluation. Findings show that adding polynomial trend and seasonal features can modestly improve predictive accuracy over simple linear models in forecasting monthly sales.


---

## Introduction

Accurate sales forecasting is essential for inventory management, workforce planning, and financial strategy in retail operations. Time series modeling leverages historical sales records, seasonal fluctuations, and categorical features (such as product family and promotions) to yield actionable predictions. This report details the workflow from dataset exploration through to predictive modeling and evaluation.

---

### Body Methods and Workflow

- **Exploratory Data Analysis (EDA):**
    - Examined the structure of sales data, including date, store number, product family, sales amounts, and promotions.
    - Assessed missing data, distribution of sales, and categorical variable representations.
    - Created monthly aggregated line plots to visualize total sales over time, revealing both trend and seasonality.
- **Feature Engineering:**
    - Extracted 'year' and 'month' values from the date field.
    - Aggregated data to monthly totals for each store/family.
    - Generated features for polynomial trend (degree=2) and encoded seasonal effects with month variables.
- **Modeling Techniques:**
    - Baseline: Linear regression on time index to predict monthly sales.
    - Enhanced: Polynomial regression (degree=2) plus seasonal dummy variables.

    - Model selection based on Root Mean Squared Error (RMSE) and R2$R$2 on held-out test data.

- **Evaluation Approach:**
    - Split dataset: 2.4 million rows for training, 600,000 rows for testing.
    - Compared sample predictions versus actuals to assess error and model fidelity.
    - Key metrics: RMSE and R2$R$2 (goodness of fit).

---

## Findings and Results

- Baseline linear model achieved an RMSE of approximately 1094.5 and R2$R$2 of 0.3537 on test data.
- Adding polynomial and seasonal features improved RMSE marginally (to 1089.9) and R2$R$2 to 0.3591.
- Prediction samples show reasonable alignment for larger sales, but occasional overestimation for periods of zero actual sales.
- The sales time series plot confirms high volatility and clear seasonal peaks, validating the need for direct modeling of seasonality.
- Feature engineering (month/year extraction, seasonal dummies) was critical to capturing recurring patterns in sales.

---

## Discussion

- The modest improvement in RMSE and R2$R$2 demonstrates that while seasonality matters, simple polynomial trends cannot alone explain all variability—potentially due to promotional effects, holidays, and unexplored categorical features.
- Some product families and promotion events could be modeled separately for finer forecasts.
- Future enhancements could include gradient boosting, ARIMA models, holiday calendars, and external regressors like oil prices.
- Data preparation—such as handling missing or zero sales, grouping by family/store, and date-based aggregation—drives reproducible and scalable modeling workflows.

---

## Conclusion

Polynomial and seasonal feature engineering improves predictive accuracy for retail sales time series compared to linear trend alone, advancing the case for structured feature addition in business forecasting. Further accuracy gains are achievable with more complex models and richer data.

---

## References

- EDA_and_Modeling.ipynb: Step-by-step notebook for analysis, feature engineering, and regression modeling.
- Source Python libraries: pandas, matplotlib, scikit-learn.
- Store sales dataset fields: date, store_nbr, family, sales, onpromotion.
- Sales aggregation and monthly trends: See attached images.
- Full code base and requirements: main.py, requirements.txt.

---