DEMAND FORECASTING PROJECT PRESENTATION

CS613: Machine Learning

Manjiri Khodke, Sriram Suvaidhar,
Khushi Patel, Mokshad Sankhe

PROBLEM STATEMENT

- The main aim is to make sure each shop receives the right amount of stock at the right time.
- We want to avoid situations where a shop runs out of products and loses customers but also prevent sending more stock than the shop can realistically sell.
- In simple terms, the objective is to maintain a healthy balance not too much, not too little so that shops can operate smoothly, meet customer demand, and avoid unnecessary costs or wastage.

DATASET

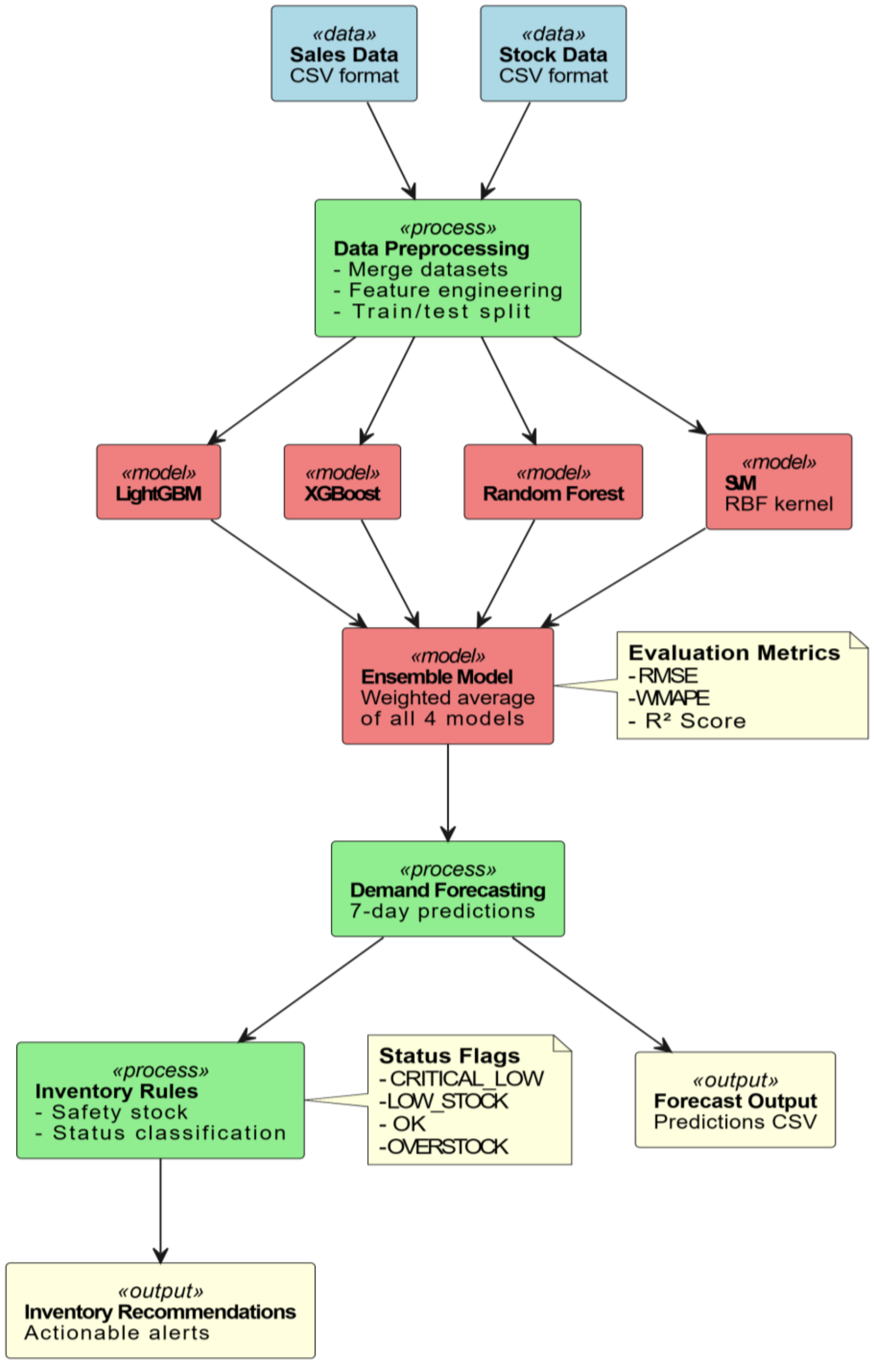
- The dataset is taken from the Google Cloud public Marketplace in Big Query.
- **Sales Data**: Daily sales transactions
  - Fields: `sales_date`, `dealer_code`, `product`, `qty`, `total_price`
- **Stock Data**: Daily inventory snapshots
  - Fields: `benchmark_date`, `ec_id`, `product_name`, `inventory_qty`
- **Scale**: Multiple stores × multiple products × daily timestamps
- **Preprocessing**: Merged datasets, handled missing values, created time-series structure.

UPGRADING STOCK FORECASTING WITH ML

- Old Method: simple average of past sales
- Limitations: Misses seasonality, trends, demand spikes, and shop differences
- New Method: ML algorithms analyzing multiple factors
- Improvement: more accurate, dynamic forecasts
- Results: better stock allocation and inventory planning
- ***Reference:***
H. Malik, "A beginner's approach to time-series with working example: Demand forecasting | Time series example with Kaggle," Medium, Jun. 11, 2024. [Online]. Available: https://medium.com/@humzahmalik/a-beginners-approach-to-time-series-with-working-example-c6bff9c24928

PROBLEM FORMULATION

- It a regression problem and not classification
- **Target Variable**: Continuous daily sales quantity (Predict how many units will sell, not just categories)
- **Why Regression?**
    - Need exact numbers for inventory ordering
    - Business decisions require precise quantities
    - Natural numeric output (0, 1, 2, ... units)



PROGRAM ARCHITECTURE

- **Data Input:** Sales and Stock data (CSV format) are fed into the system.
- **Data Preprocessing:** The data is merged, features are engineered, and it's split for training/testing.
- **Modeling:** Four machine learning models (LightGBM, XGBoost, Random Forest, SVR) are trained, and an Ensemble Model is created from their weighted average.
- **Evaluation:** The models are assessed using metrics like RMSE, WMAPE, and R^2 Score.
- **Demand Forecasting:** The Ensemble Model is used to make 7-day predictions.
- **Inventory Management:** The forecasts are processed through Inventory Rules (e.g., safety stock) to generate Status Flags (CRITICAL LOW, LOW STOCK, OK, OVERSTOCK).
- **Output:** The system produces a Forecast Output CSV (the predictions) and Inventory Recommendations (actionable alerts) based on the status flags.

LIGHT GRADIENT BOOSTING MACHINE (LIGHTGBM)

- Light Gradient Boosting Machine (LightGBM) serves as our primary model, selected for its superior speed and predictive performance.
- It is a gradient boosting framework that builds trees in a leaf-wise manner, which often achieves higher accuracy compared to traditional level-wise growth.
- The model accelerates training on large datasets through a histogram-based algorithm for efficient feature splitting and supports categorical variables natively without preprocessing.
- Furthermore, its compatibility with GPU execution ensures scalability and production readiness for high-performance deployments.

SUPPORT VECTOR REGRESSION (SVR)

- It is a Kernel-Based Approach for Regression.
- This method utilizes Support Vector Regression (SVR), which adapts the principles of Support Vector Machines (SVM) from classification to predict continuous target values.
- The approach employs the kernel trick to implicitly map input data into higher-dimensional spaces, enabling the modeling of complex, non-linear relationships. Our experimentation tested multiple configurations, including the Radial Basis Function (RBF) kernel with various regularization (C) and epsilon parameters, alongside linear and polynomial kernels for comparative analysis.
- Given SVR's sensitivity to feature scale, all inputs were normalized using a StandardScaler prior to model training.

COMPLEMENTARY MODELS

- **XGBoost**
  - Similar to LightGBM but uses level-wise tree growth
  - Additional regularization terms
  - Early stopping based on validation RMSE
- **Random Forest**
  - Ensemble of decision trees
  - Bagging (bootstrap aggregation) approach
  - Built-in feature importance

ENSEMBLE STRATEGY

- It is a Weighted Average Ensemble Approach.
- **Rationale**: Combine strengths, mitigate weaknesses
- **Weight Calculation**: Inverse of RMSE (better models get higher weight)

$$w_i = \frac{1/\mathrm{RMSE}_i}{\sum_{j=1}^{4}(1/\mathrm{RMSE}_j)}$$

- **Final Prediction**:

$$\hat{y} = w_{\mathrm{lgb}} \cdot \hat{y}_{\mathrm{lgb}} + w_{\mathrm{xgb}} \cdot \hat{y}_{\mathrm{xgb}} + w_{\mathrm{rf}} \cdot \hat{y}_{\mathrm{rf}} + w_{\mathrm{svm}} \cdot \hat{y}_{\mathrm{svm}}$$

FEATURE ENGINEERING STRATEGY

- The strategy focuses on Creating Predictive Signals by extracting and synthesizing information across multiple domains.
- Temporal features are generated from dates, including day, month, quarter, and weekend indicators.
- Historical purchase patterns are captured through lagged variables, such as 1-day and 7-day lags, and rolling averages.
- To model store-product specificity, unique identifiers are encoded, and historical averages and standard deviations are calculated per combination.
- Additional context is incorporated from inventory data, including ending stock levels and derived turnover ratios, alongside price signals based on average unit cost calculations.

TRAIN-TEST METHODOLOGY

- The model development process employs a rigorous time-based splitting strategy.
- A chronological split is critical to prevent data leakage and simulate real-world forecasting conditions.
- Data is partitioned with an 80% training and 20% testing ratio, strictly maintaining temporal order.
- Validation is conducted using a subset of the training data for early stopping to avoid overfitting.
- To ensure a fair and consistent evaluation across all models, the same fixed test set is used for all performance comparisons.

EVALUATION METRICS

- **RMSE** (Root Mean Square Error):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  - RMSE is the main accuracy score where larger errors count more heavily toward the final result.
- **WMAPE** (Weighted Mean Absolute Percentage Error):

$$\text{WMAPE} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n} y_i}$$

  - WMAPE translates errors into an easy-to-understand percentage based on total sales
- **MAE** (Mean Absolute Error):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  - MAE gives the straightforward average of all errors, treating each mistake the same regardless of size.

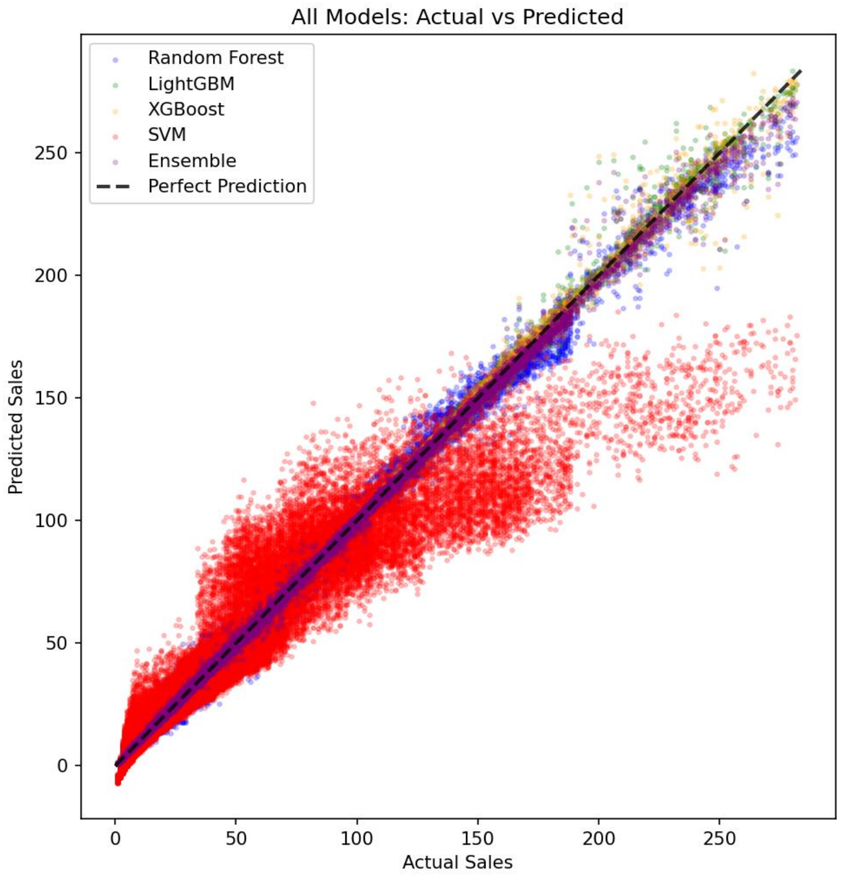
RESULTS - MODEL PERFORMANCE

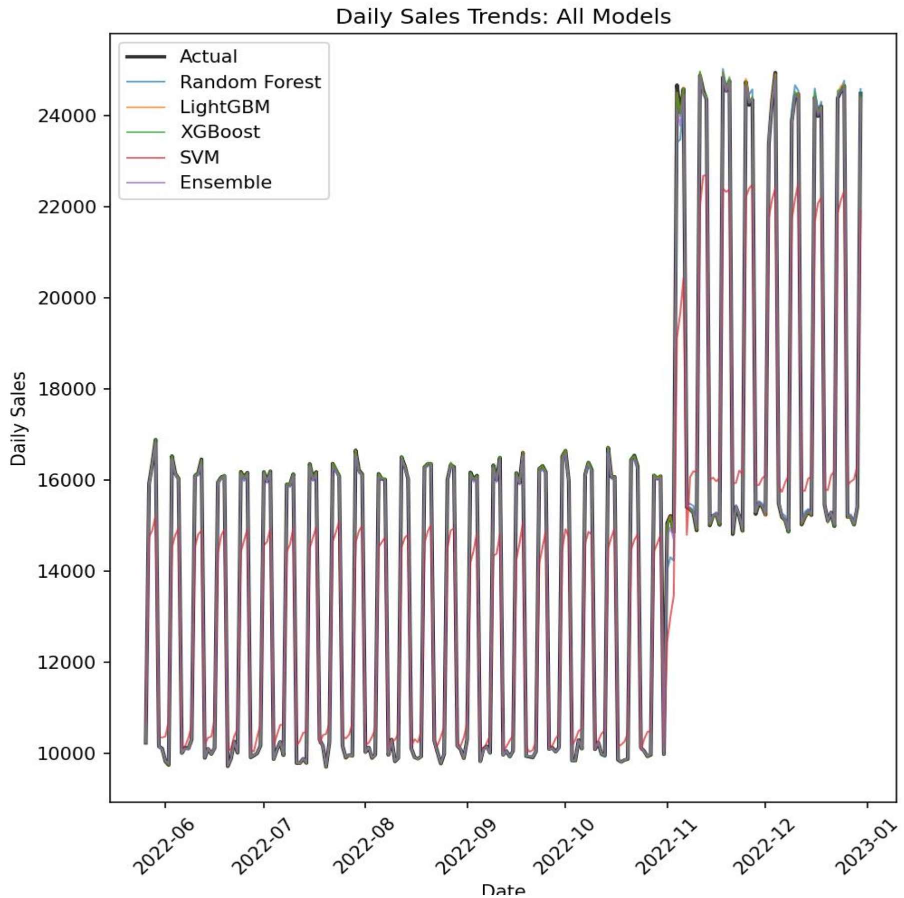
Model	RMSE ↓	MAE ↓	WMAPE ↓
LightGBM	0.9573	0.3531	1.09%
XGBoost	1.0774	0.4227	1.30%
Random Forest	2.3733	1.2565	3.86%
SVM (RBF)	12.0908	6.1007	18.76%
ENSEMBLE	1.0383	0.4559	1.40%

RESULTS - FEATURE IMPORTANCE

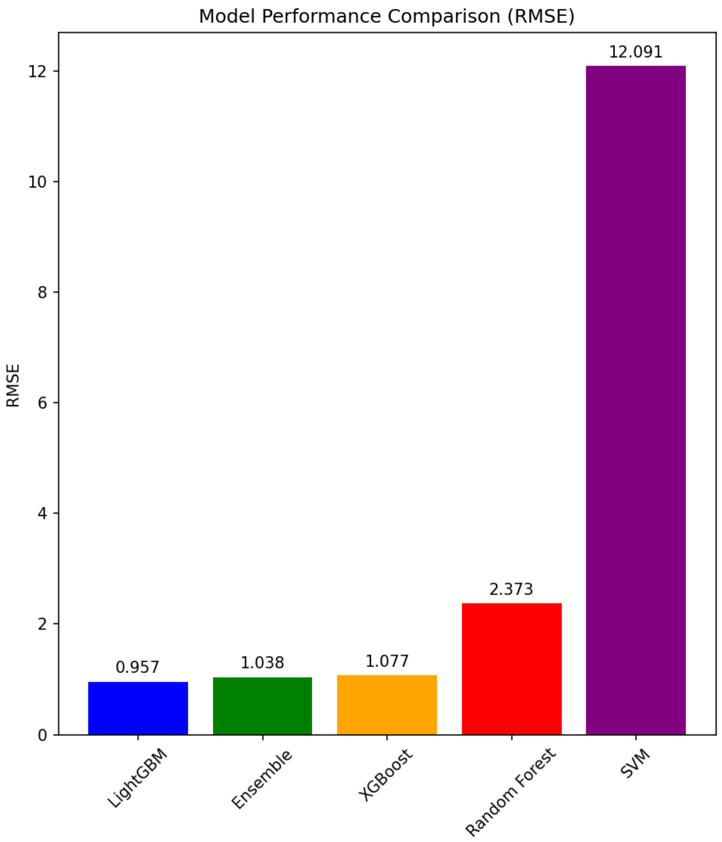
- Analysis of our LightGBM model reveals the key drivers of predictions.
- Top 5 Features (LightGBM Gain-based):
  - sales_lag_1 - Yesterday's sales (strongest predictor)
  - store_product_avg_sales - Historical baseline
  - sales_rolling_7 - Weekly trend
  - ending_inventory - Current stock levels
  - day_of_week - Weekly seasonality
- This ranking shows that immediate recent history and store-product specific patterns are the most influential factors in our forecasts.

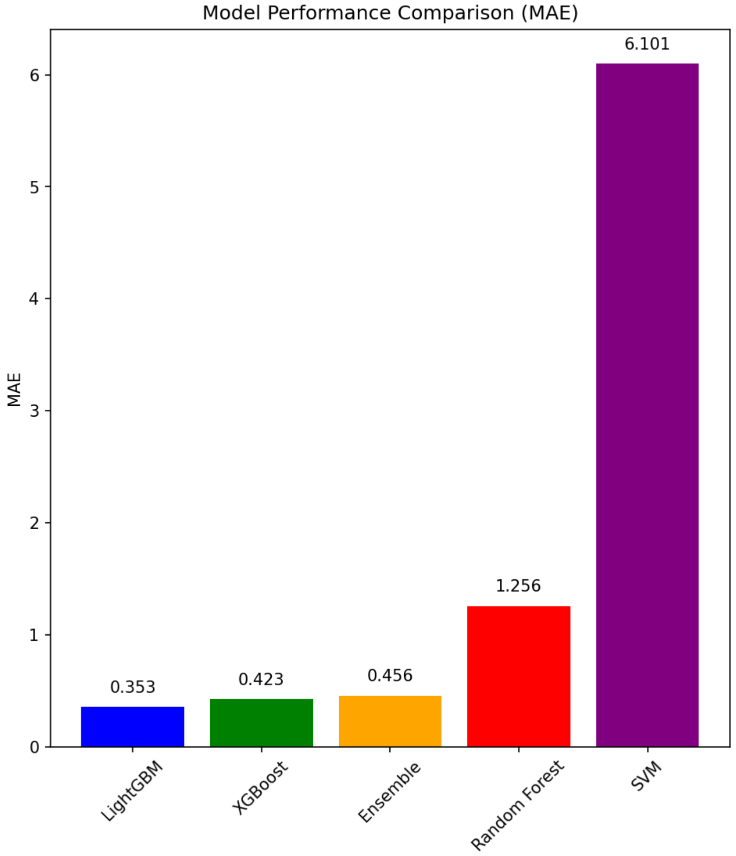
VISUALIZATION

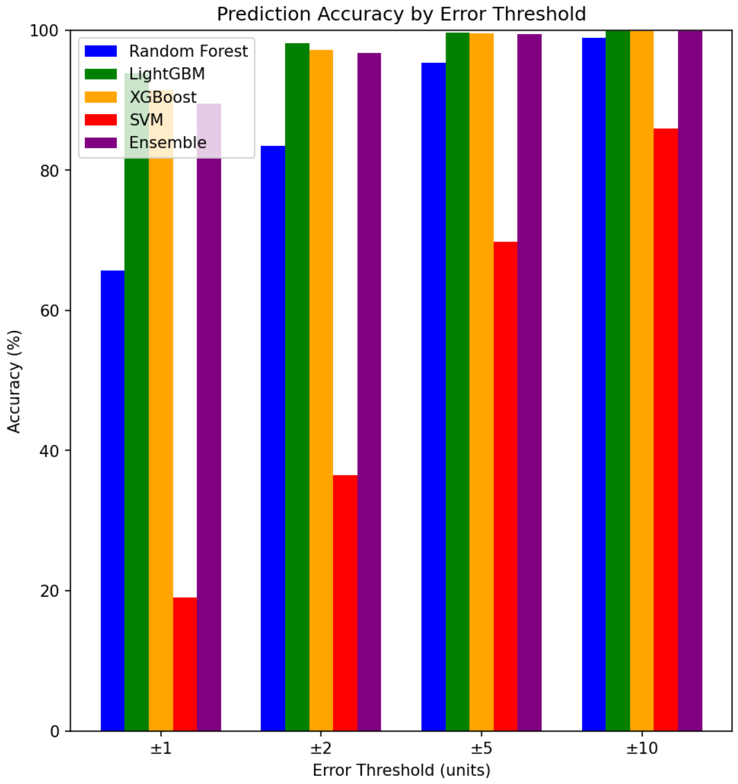




VISUALIZATION







FUTURE EXTENSIONS

- **External Factors Integration**:
  - Holiday calendars
  - Promotional events
  - Weather data
- **Advanced Techniques**:
  - Deep learning (LSTM/GRU for sequence modeling)
  - Bayesian optimization for hyperparameter tuning
- **System Improvements**:
  - Real-time updates with streaming data
  - Automated retraining pipeline
  - A/B testing framework for model updates

CONCLUSION

- This project shows that machine learning can turn inventory management into a smarter, proactive process.
- By using four different models together, we built a system that forecasts daily sales with high accuracy.
- Our solution provides actionable restocking suggestions, helping businesses reduce both missed sales from stockouts and waste from overstock.

THANK YOU! QUESTIONS?