Rossmann Sales Forecasting

Demand Modeling

Predict product demand per store

Time Series

Handle seasonality and trends

Promotions

Account for campaign effects

Store Clustering

Group similar locations

External Data

Incorporate weather & events

Model Ensemble

Combine multiple algorithms

Evaluation

Track accuracy and bias

Deployment

Automate forecasts to stores

NEXT HIKES
IT SOLUTIONS

Internship: Nexthikes IT Solution

Presented By: Deepa Pathak

Data at a Glance: Fueling Our Predictions

Our forecasting model was built upon three distinct, yet interconnected, datasets provided by Rossmann. Understanding their structure and volume was the first crucial step.

1

Train.csv

Over 1 Million Records: This extensive dataset provided historical daily sales data, including store details, promotion status, and customer counts. It was the backbone for model training.

2

Test.csv

41,000+ Records: This dataset contained the future dates for which sales predictions were required. It mirrored the structure of the training data but lacked the actual sales figures.

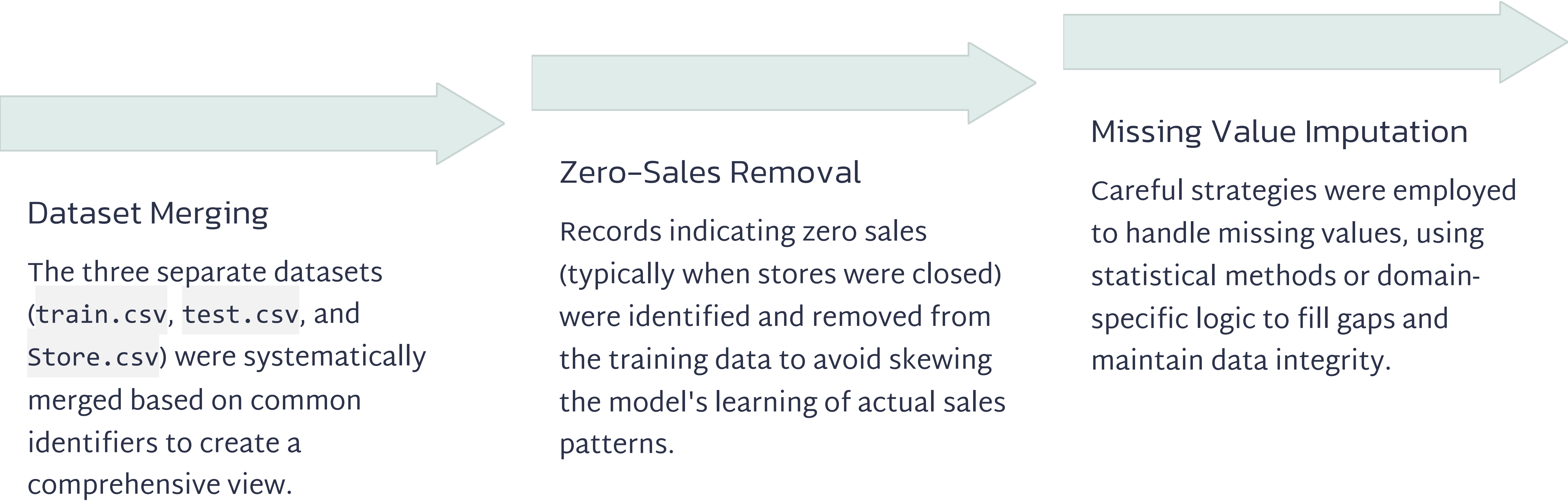
3

Store.csv

Store-Specific Details: A supplementary dataset providing static information about each store, such as store type, assortment, nearest competition distance, and ongoing promotions.

Data Preparation: Laying the Foundation for Accuracy

Raw data, no matter how vast, requires meticulous cleaning and structuring to be valuable for machine learning. Our data preparation phase involved several critical steps to ensure data quality and model readiness.



Dataset Merging

The three separate datasets (`train.csv`, `test.csv`, and `Store.csv`) were systematically merged based on common identifiers to create a comprehensive view.

Zero-Sales Removal

Records indicating zero sales (typically when stores were closed) were identified and removed from the training data to avoid skewing the model's learning of actual sales patterns.

Missing Value Imputation

Careful strategies were employed to handle missing values, using statistical methods or domain-specific logic to fill gaps and maintain data integrity.

Feature Engineering: Unlocking Deeper Insights

Beyond the raw data, we engineered new features that captured temporal patterns and competitive dynamics. These derived features significantly enhanced the model's ability to identify underlying sales drivers.

Year, Month, WeekOfYear

Extracted from the Date column, these features captured yearly trends, monthly seasonality, and weekly sales cycles (e.g., weekends vs. weekdays).

CompetitionOpen

Calculated the number of days a competitor store had been open nearby, providing insight into competitive pressure over time.

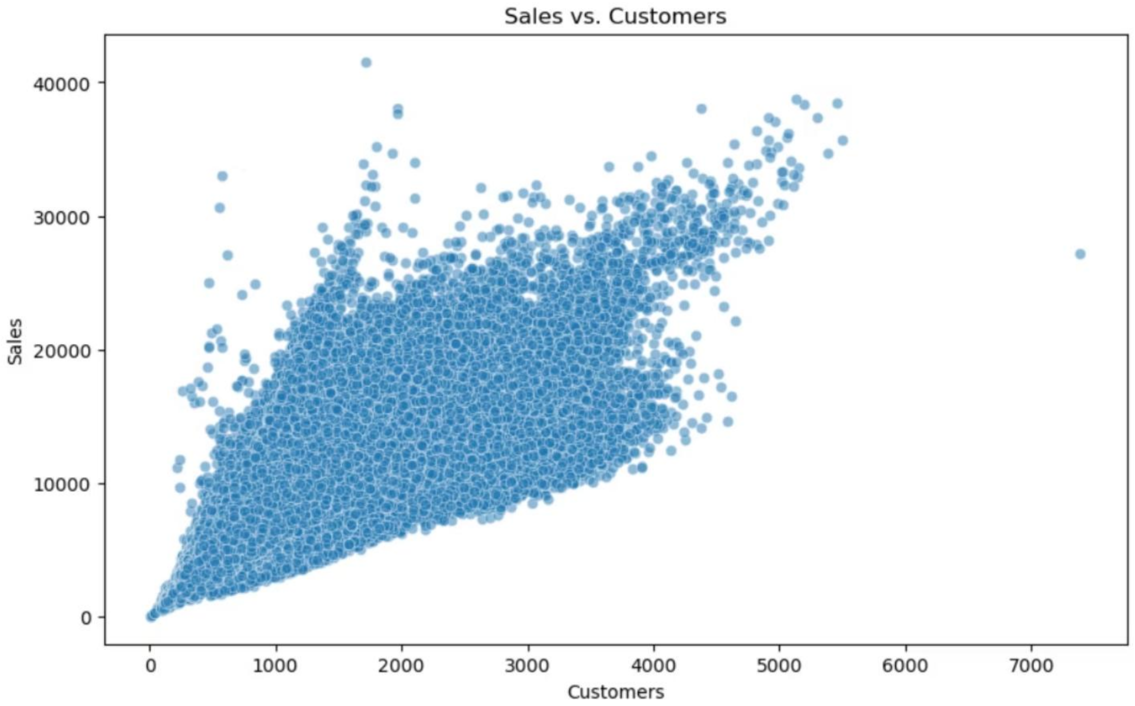
Promo2Open

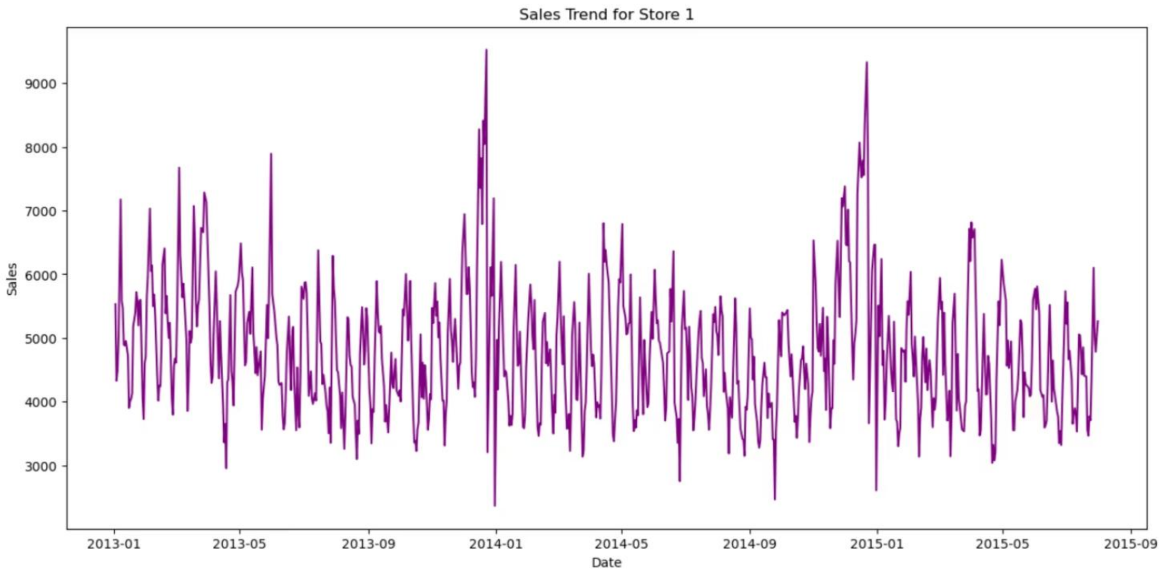
Determined the number of days a store had been participating in a continuous promotion, indicating the duration and potential impact of ongoing marketing efforts.

Future Enhancements

Consideration for external factors like local holidays, school breaks, and even weather patterns were noted for future feature additions.

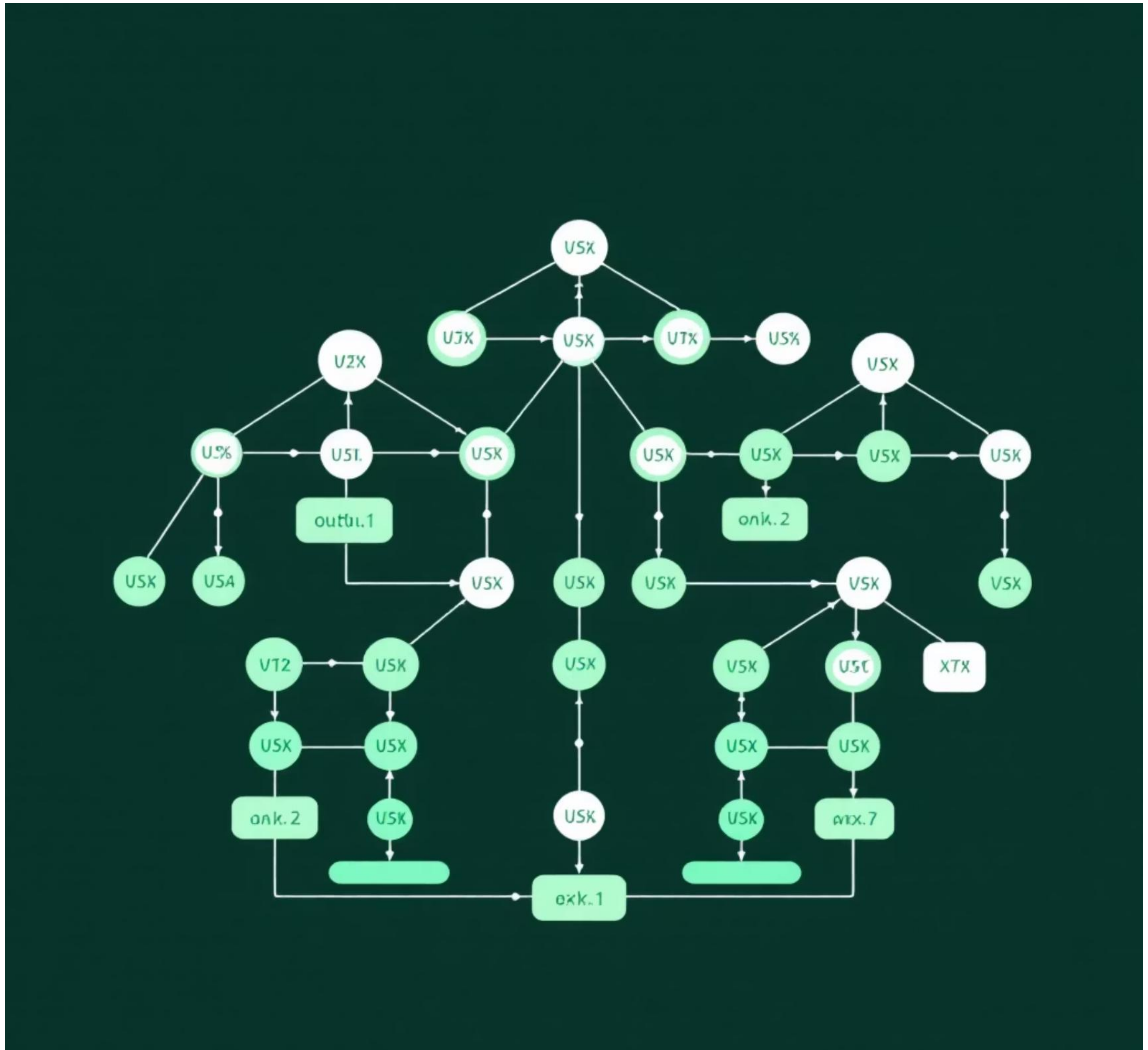
Graph


Sales vs. Customers
40000
30000
20000
10000
0
Sales
0
1000
2000
3000
4000
5000
6000
7000
Customers


Sales Trend for Store 1
9000
8000
7000
6000
5000
4000
3000
Sales
2013-01
2013-05
2013-09
2014-01
2014-05
2014-09
2015-01
2015-05
2015-09
Date

Model Selection: The Power of XGBoost Regressor

For this complex forecasting challenge, we chose the XGBoost Regressor. Its robust architecture and proven performance make it an ideal choice for problems with intricate, non-linear relationships and diverse feature sets.



Why XGBoost?

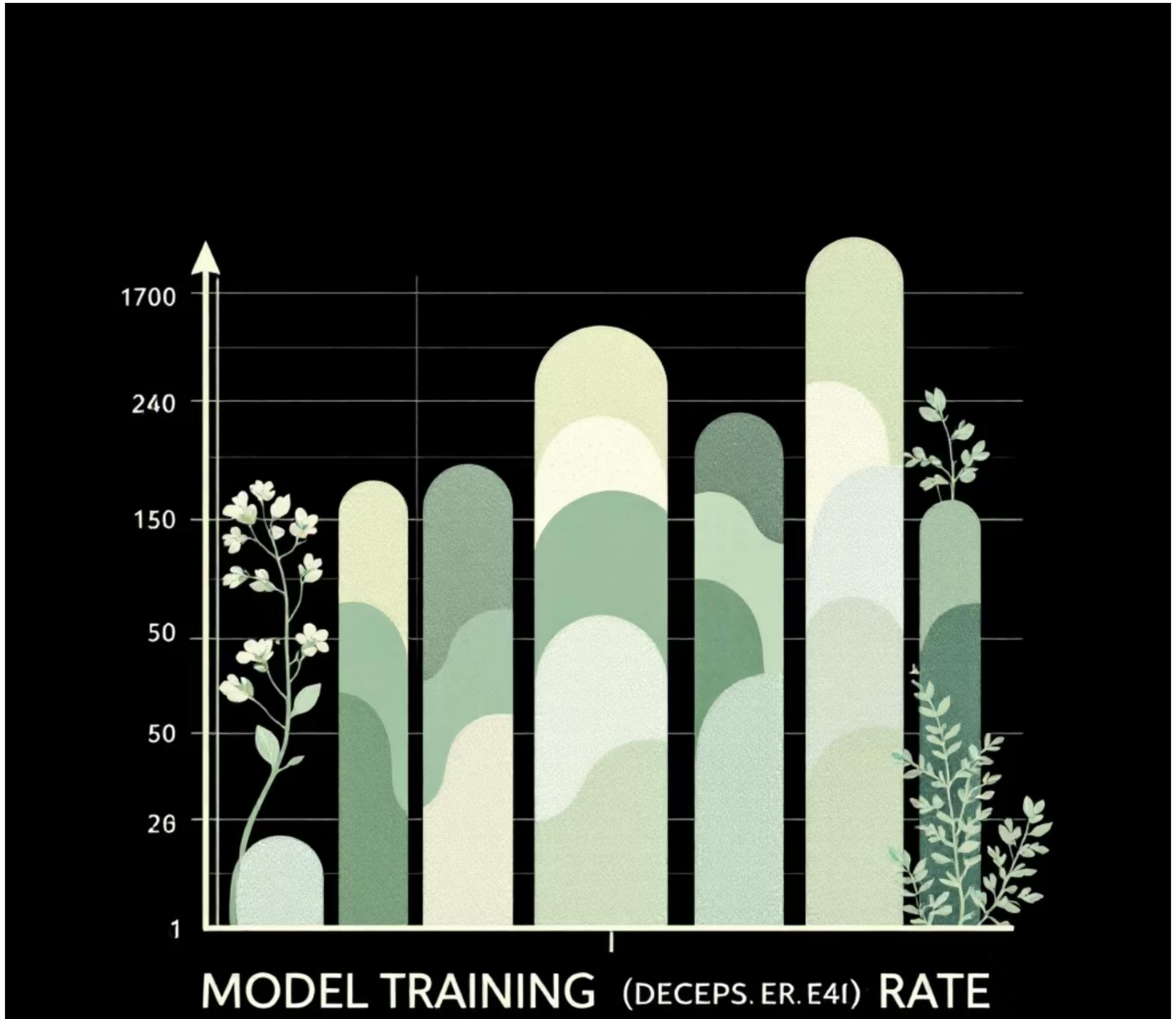
- **Handles Non-Linearity:** Effectively captures complex, non-linear relationships between various features (e.g., promotions, competition, time of year) and sales.
- **High Performance:** Known for its speed and accuracy, often outperforming other algorithms on structured data.
- **Feature Importance:** Provides insights into which features contribute most to predictions, aiding in business understanding.
- **Regularization:** Built-in regularization helps prevent overfitting, leading to more generalized and reliable forecasts.

Model Training & Evaluation: Proving Predictive Power

With clean data and engineered features, the XGBoost model was trained. Evaluating its performance rigorously was key to ensuring its reliability for real-world application.

Training Process

The model ingested the prepared historical data, learning the intricate patterns and relationships that drive sales. This involved an iterative process of adjusting internal parameters to minimize prediction errors.



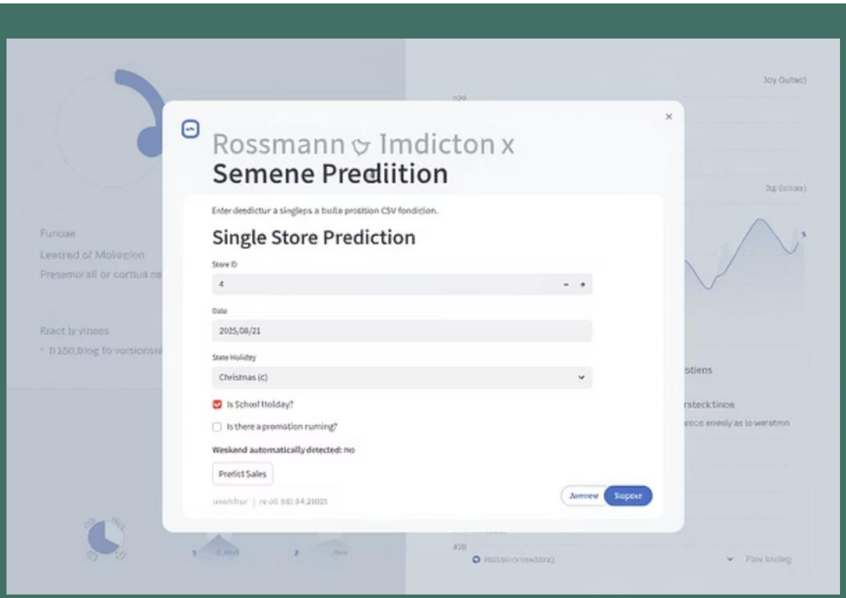
Evaluation Metric: RMSPE

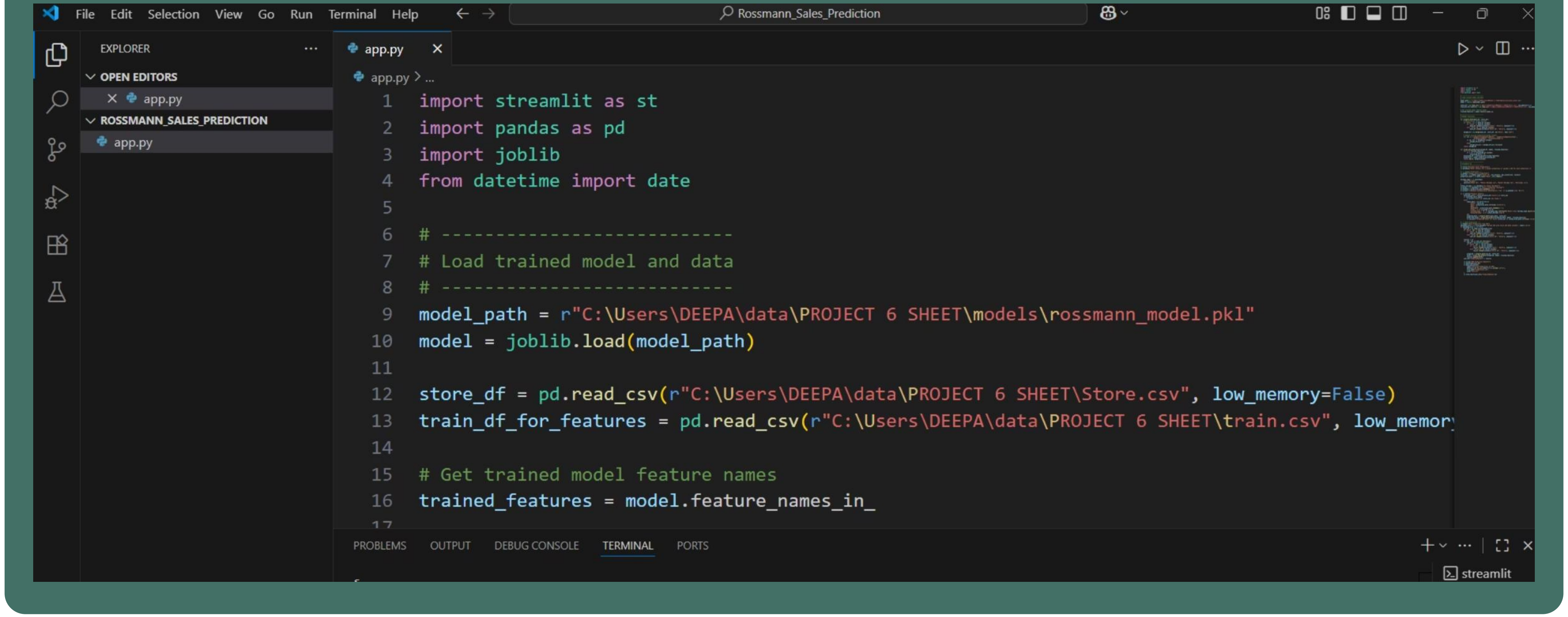
We primarily used the **Root Mean Square Percentage Error (RMSPE)** to evaluate the model. RMSPE is particularly suitable for sales forecasting as it penalizes larger percentage errors more, giving a clear indication of prediction accuracy relative to actual sales values.

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

Where *y_ i* is the actual sale and *|hat{y}_ i* is the predicted sale.

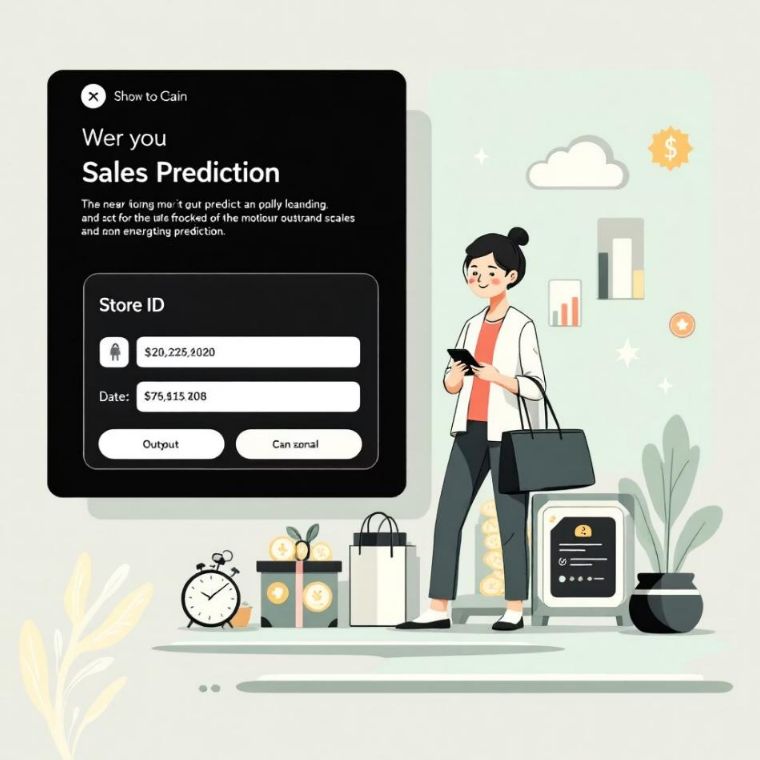
Deployment: From Model to Real-Time Predictions




File Edit Selection View Go Run Terminal Help
Rossmann_Sales_Prediction
EXPLORER
OPEN EDITORS
app.py
ROSSMANN_SALES_PREDICTION
app.py
app.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
streamlit

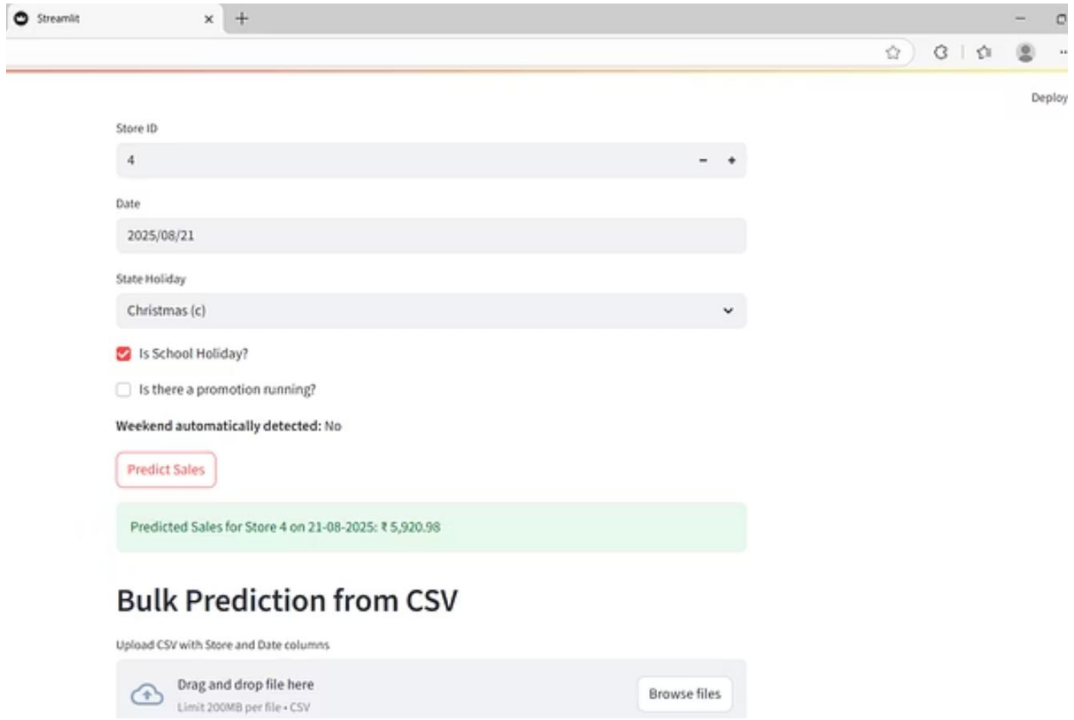
```python
import streamlit as st
import pandas as pd
import joblib
from datetime import date

# ---------------------------
# Load trained model and data
# ---------------------------
model_path = r"C:\Users\DEEPA\data\PROJECT 6 SHEET\models\rossmann_model.pkl"
model = joblib.load(model_path)

store_df = pd.read_csv(r"C:\Users\DEEPA\data\PROJECT 6 SHEET\Store.csv", low_memory=False)
train_df_for_features = pd.read_csv(r"C:\Users\DEEPA\data\PROJECT 6 SHEET\train.csv", low_memor

# Get trained model feature names
trained_features = model.feature_names_in_
```

- **Interactive Interface:** Users can select specific dates and store IDs to get instant sales predictions.
- **Trained Model Loading:** The application efficiently loads the pre-trained XGBoost model, ensuring quick response times.
- **Real-time Forecasts:** Provides immediate insights into projected sales, supporting dynamic decision-making



Deployment: From Model to Real-Time Predictions

To make the forecasting solution accessible and actionable for Rossmann management, we developed an intuitive web application using Stream lit.



Store ID

4

Date

2025/08/21

State Holiday

Christmas (c)

☑ Is School Holiday?

□ Is there a promotion running?

Weekend automatically detected: No

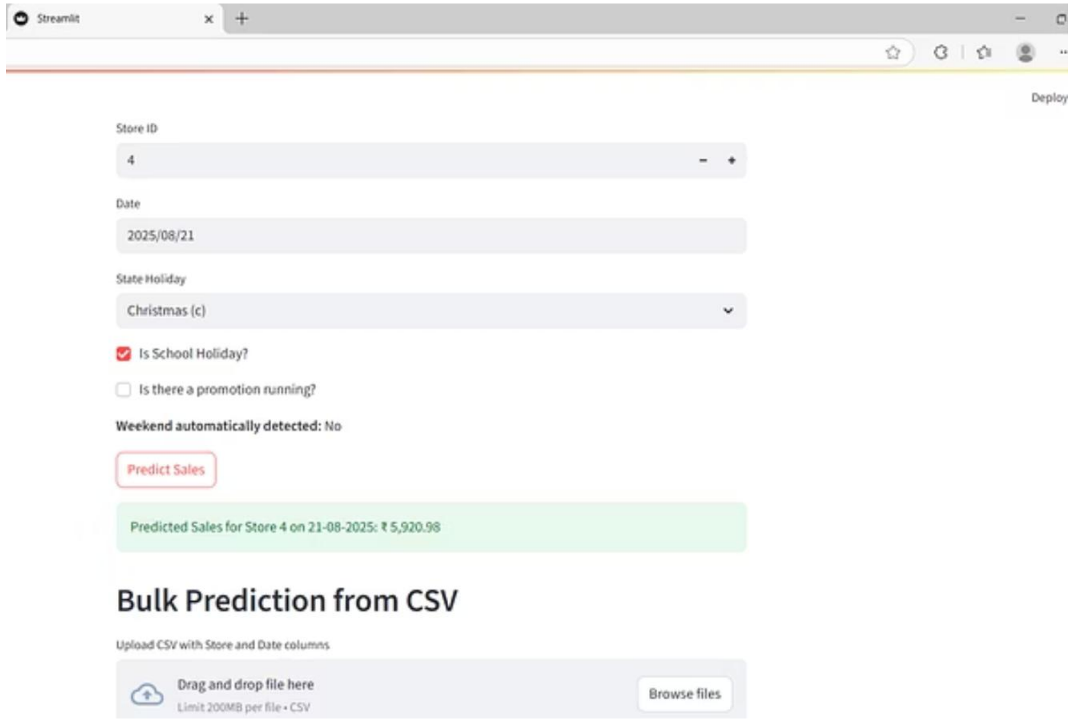
Predicted Sales for Store 4 on 21-08-2025: ₹ 5,920.98

Bulk Prediction from CSV

Upload CSV with Store and Date columns

Drag and drop file here

Limit 200MB per file • CSV


Streamlit
Deploy
Store ID
4
Date
2025/08/21
State Holiday
Christmas (c)
☑ Is School Holiday?
□ Is there a promotion running?
Weekend automatically detected: No
Predict Sales
Predicted Sales for Store 4 on 21-08-2025: ₹ 5,920.98
Bulk Prediction from CSV
Upload CSV with Store and Date columns
Drag and drop file here
Limit 200MB per file • CSV
Browse files

Future Work: Continuous Improvement & Expansion

The current forecasting solution provides a strong foundation, but the journey of improvement is continuous. Here are key areas for future development:

Hyperparameter Tuning

Fine-tuning the XGBoost model's hyperparameters using techniques like GridSearchCV or RandomizedSearchCV to further optimize performance.

Advanced Feature Engineering

Integrating external data (e.g., local events, weather, economic indicators) and creating more complex interaction features to capture nuanced sales drivers.

Model Stacking/Ensembling

Combining predictions from multiple diverse models (e.g., ARIMA, LightGBM, Neural Networks) to potentially achieve higher accuracy and robustness.

Factors Influencing Rossmann's Sales & Feedback 👇

- Based on the data analysis, sales are significantly higher during promotions and are directly influenced by store type and competition distance. Sales predictably drop on holidays and show clear seasonal and weekly patterns.

- Positive Feedback: The project successfully delivers on the core objective of sales forecasting. The use of a powerful model like XG Boost ensures high accuracy, and the Stream lit app makes the predictions easily accessible. The end product is a valuable asset that can be integrated into the finance team's planning process.
- Negative Feedback & Next Steps: For a more advanced and accurate solution, we recommend further investment in two key areas. First, we could explore **Deep Learning models**, such as LSTM networks, which are specifically designed for time-series data and can capture more complex patterns. Second, implementing **MLOps tools** like DVC and MLFlow would allow us to track and manage different data and model versions, ensuring the project is scalable and maintainable for the long term.

Thank You!

Git hub: DataWithDeepa (Deepa Pathak)

Email ID: deepapathak121@gmail.com