Marketing Mix Modeling (MMM) for Budget Optimization

> **An end-to-end Marketing Mix Model using Linear Regression to quantify marketing channel impact, simulate budget scenarios, and serve predictions via a Flask REST API.**

Project Overview

This project delivers a **production-ready Marketing Mix Model** that:

- **Quantifies** the revenue impact of digital marketing channels
- **Calculates** ROI across Google, Meta, and organic channels
- **Simulates** "What-If" budget reallocation scenarios
- **Serves** predictions via a lightweight Flask REST API
- **Visualizes** performance trends and feature importance

Repository Structure

```
Marketing-Mix-Modeling-MMM-for-Marketing-Budget-Optimization-main/
│
├── ecommerce_mmm_model_training.ipynb  # Full training pipeline (47 cells)
├── mmm_app.py                          # Flask API server
├── mmm_model_features.json             # Feature list (12 features)
├── README.md                           # Original project readme
└── my_readme.md                        # This comprehensive guide
```

Note: The trained model `linear_mmm_model.pkl` is generated after running the notebook.

Dataset Description

Source: Multi-Region Ecommerce MMM Dataset

Metric	Value
Total Rows	132,759
Total Columns	50 (original) → 35 (after cleaning)
Time Range	Daily data through 2024
Train Period	≤ 2023-12-31

Metric	Value
Test Period	> 2023-12-31 (2024+)

Target Variable

Variable	Calculation	Description
`revenue`	`ALL_PURCHASES_ORIGINAL_PRICE - ALL_PURCHASES_GROSS_DISCOUNT`	Net sales revenue

Input Features

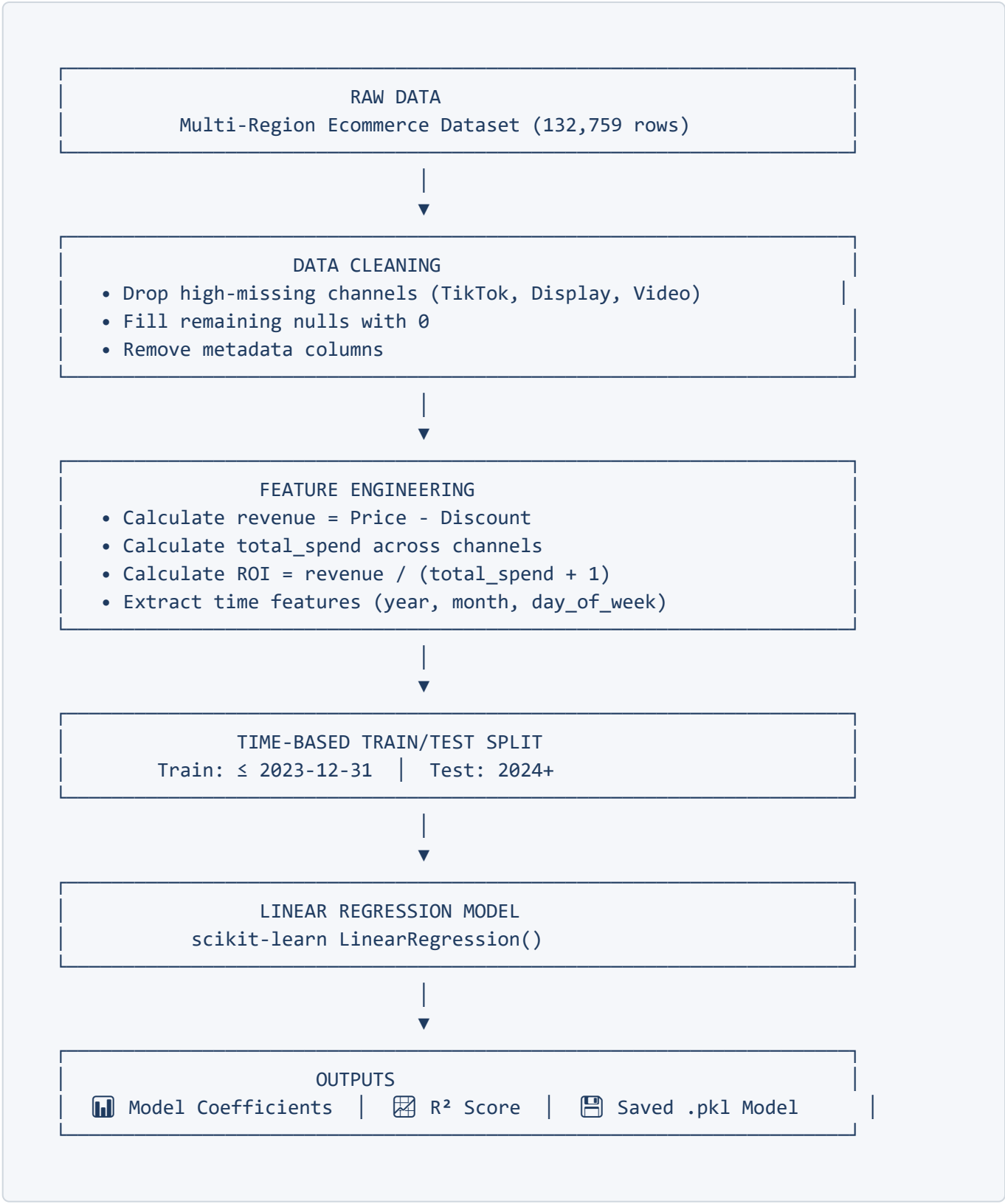
Category	Feature	Description
Google Ads	`GOOGLE_PAID_SEARCH_SPEND`	Search advertising spend ($)
	`GOOGLE_SHOPPING_SPEND`	Shopping campaign spend ($)
	`GOOGLE_PMAX_SPEND`	Performance Max spend ($)
Meta Ads	`META_FACEBOOK_SPEND`	Facebook advertising spend ($)
	`META_INSTAGRAM_SPEND`	Instagram advertising spend ($)
Organic	`EMAIL_CLICKS`	Email marketing clicks
	`ORGANIC_SEARCH_CLICKS`	Organic search traffic clicks
	`DIRECT_CLICKS`	Direct website visits
	`BRANDED_SEARCH_CLICKS`	Brand term search clicks
Temporal	`year`	Year component
	`month`	Month (1-12)
	`day_of_week`	Day of week (0-6)

Dropped Channels (High Missing Data)

```
❌ TikTok (Spend, Clicks, Impressions)
❌ Google Video (Spend, Clicks, Impressions)
❌ Google Display (Spend, Clicks, Impressions)
❌ Meta Other (Spend, Clicks, Impressions)
```

Methodology

End-to-End Pipeline



Why Linear Regression?

Linear Regression is the **standard baseline for MMM** because:

Advantage	Explanation
☑ **Interpretable**	Coefficients = marginal revenue contribution
☑ **Fast Training**	Handles 100K+ rows efficiently

Advantage	Explanation
☑ **No Hyperparameters**	Deterministic results
☑ **Business Aligned**	Coefficients map directly to ROI

Model Performance

Evaluation Metrics

Metric	Value	Interpretation
R^2 Score	`0.789`	78.9% variance explained
RMSE	`246,748,035,379.62`	Average prediction error (in revenue units)

Feature Coefficients (ROI Signals)

Feature	Coefficient	Direction
year	+19,603.65	Positive
month	+2,600.52	Positive
GOOGLE_PAID_SEARCH_SPEND	+86.76	Positive
EMAIL_CLICKS	+80.48	Positive
GOOGLE_SHOPPING_SPEND	+46.87	Positive
META_INSTAGRAM_SPEND	+14.36	Positive
META_FACEBOOK_SPEND	+7.99	Positive
BRANDED_SEARCH_CLICKS	+5.15	Positive
DIRECT_CLICKS	-19.45	Negative
GOOGLE_PMAX_SPEND	-23.38	Negative
day_of_week	-1,347.36	Negative

Coefficient Interpretation

Coefficient	Meaning
+86.76 (Google Paid Search)	Every $1 spent → $86.76 revenue increase
+80.48 (Email Clicks)	Each click → $80.48 revenue
-23.38 (Google PMax)	Negative ROI — may need optimization

What-If Simulation

Scenario: +30% Google Paid Search Spend

```
scenario = X_test.copy()
scenario['GOOGLE_PAID_SEARCH_SPEND'] *= 1.3  # +30% increase

y_simulated = model.predict(scenario)
change = ((y_simulated.mean() - y_pred.mean()) / y_pred.mean()) * 100
```

Result: +4.60% simulated revenue increase

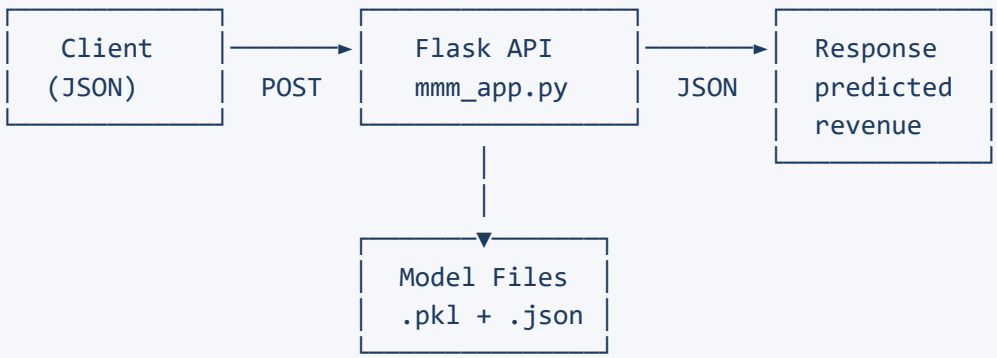
Other Scenarios to Explore

Scenario	Code Modification
Cut Meta Facebook by 50%	`scenario['META_FACEBOOK_SPEND'] *= 0.5`
Double Email Clicks	`scenario['EMAIL_CLICKS'] *= 2`
Shift $10K from PMax to Shopping	Subtract from PMax, add to Shopping

---

Flask API

Architecture



Endpoints

Method	Endpoint	Description
`GET`	`/`	Health check
`POST`	`/predict`	Revenue prediction

API Usage

Start Server

```
python mmm_app.py
# Server runs at http://127.0.0.1:5000
```

Health Check

```
curl http://127.0.0.1:5000/
# Response: "MMM Model is running!"
```

Prediction Request

```
curl -X POST http://127.0.0.1:5000/predict \
  -H "Content-Type: application/json" \
  -d '{
    "GOOGLE_PAID_SEARCH_SPEND": 100000,
    "GOOGLE_SHOPPING_SPEND": 50000,
    "GOOGLE_PMAX_SPEND": 20000,
    "META_FACEBOOK_SPEND": 30000,
    "META_INSTAGRAM_SPEND": 15000,
    "EMAIL_CLICKS": 5000,
    "ORGANIC_SEARCH_CLICKS": 8000,
    "DIRECT_CLICKS": 4000,
    "BRANDED_SEARCH_CLICKS": 6000,
    "year": 2024,
    "month": 5,
    "day_of_week": 2
  }'
```

Response

```
{
  "predicted_revenue": 123456789.0
}
```

🚀 Quick Start

Option 1: Run Notebook

```
# Clone repository
git clone <repository-url>
cd Marketing-Mix-Modeling-MMM-for-Marketing-Budget-Optimization-main

# Create virtual environment
python -m venv mmm_env
source mmm_env/bin/activate  # Windows: mmm_env\Scripts\activate

# Install dependencies
pip install pandas numpy seaborn matplotlib scikit-learn flask joblib

# Launch notebook
jupyter notebook ecommerce_mmm_model_training.ipynb
```

Option 2: Run API Server

```
# After training the model via notebook
python mmm_app.py
```

Dependencies

```
pandas>=1.3.0
numpy>=1.21.0
seaborn>=0.11.0
matplotlib>=3.4.0
scikit-learn>=1.0.0
flask>=2.0.0
joblib>=1.1.0
```

Notebook Workflow

Step	Cell Range	Description	Key Output
1	Import & Load	Load libraries and dataset	`df` (132,759 × 50)
2	Missing Values	Drop/fill missing data	Clean `df`
3	Feature Engineering	Create revenue, ROI, time features	`revenue`, `roi`, `year`, `month`
4	Train-Test Split	Chronological split at 2023-12-31	`X_train`, `X_test`, `y_train`, `y_test`

Step	Cell Range	Description	Key Output
5	Model Training	Fit LinearRegression	`model`
6	Evaluation	Calculate R^2, RMSE	Performance metrics
7	Feature Importance	Extract coefficients	Channel ROI signals
8	Simulation	+30% Google Paid Search	+4.60% revenue lift
9	Save Model	Export .pkl and .json	`linear_mmm_model.pkl`

📊 Visualizations

1. Sales Over Time

Time series plot showing revenue trends across the dataset period.

2. ROI Over Time

Dynamic ROI tracking: `revenue / (total_spend + 1)`

3. Actual vs Predicted Revenue

Model fit visualization comparing ground truth vs predictions on test set.

💡 Business Applications

Channel Optimization Matrix

Channel	Coefficient	Action
Google Paid Search	+86.76	☑ **Increase budget**
Email Clicks	+80.48	☑ **Scale email campaigns**
Google Shopping	+46.87	☑ **Maintain/increase**
Meta Instagram	+14.36	⚠ **Monitor ROI**
Meta Facebook	+7.99	⚠ **Optimize creative**
Google PMax	-23.38	❌ **Review/reduce**

Budget Reallocation Strategy

```
Current State              Recommended Action
─────────────              ──────────────────
Google PMax: $50K ──────► Reduce to $30K (-40%)
Google Search: $80K ─────► Increase to $100K (+25%)
```

```
Email Marketing: Low ──────► Double investment
Meta: Flat ──────────────────► Test creative variations
```

⚠️ Limitations & Assumptions

Limitation	Implication
Linear relationships only	Doesn't capture saturation or carryover
No lag effects	Ad impact assumed immediate
Static coefficients	Channel effectiveness may vary over time
Missing external factors	Competitor activity, seasonality shocks not modeled
Large RMSE	High variance in revenue — consider log transform

🔮 Future Improvements

Enhancement	Description
🎛️ **Regularization**	Add Ridge, Lasso, ElasticNet for stability
🔄 **Adstock/Carryover**	Model lagged marketing effects
📉 **Saturation Curves**	Diminishing returns modeling
🚀 **Production Deploy**	Docker, Gunicorn, Heroku/Render
📊 **Dashboard**	Streamlit/Dash for interactive simulations
🤖 **AutoML**	XGBoost, LightGBM, or Prophet integration

🔧 Troubleshooting

Issue	Solution
`scikit-learn` version warning	Retrain model in current environment
Flask `Address already in use`	Change port: `app.run(port=5001)`
Missing `linear_mmm_model.pkl`	Run notebook first to generate model
JSON decode error in API	Ensure valid JSON with correct feature names
High RMSE	Apply log transform to revenue

🧰 Tech Stack

Category	Technologies

Category	Technologies
Language	Python 3.8+
Data Processing	pandas, numpy
Visualization	seaborn, matplotlib
Machine Learning	scikit-learn
API Framework	Flask
Serialization	joblib, json

References

- scikit-learn Linear Regression
- Flask Documentation
- Marketing Mix Modeling Overview
- Google's MMM Methodology

Credits

Project Source: Marketing Mix Modeling for E-commerce Budget Optimization

---

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/enhancement`)
3. Commit changes (`git commit -m 'Add enhancement'`)
4. Push to branch (`git push origin feature/enhancement`)
5. Open a Pull Request

Star this repo if you find it useful!

Made with for Marketing Analytics