

DS Time-Series Project

Forecasting Daily Grocery Sales

Forecast daily grocery sales demand in Ecuador

Why is it important?

- Plan inventory more effectively
- Reduce waste
- Avoid stockouts
- Improve operational efficiency

Main Objective

build a model to predict *daily grocery sales demand* as accurately as possible



Dataset Overview

Where is the data from?

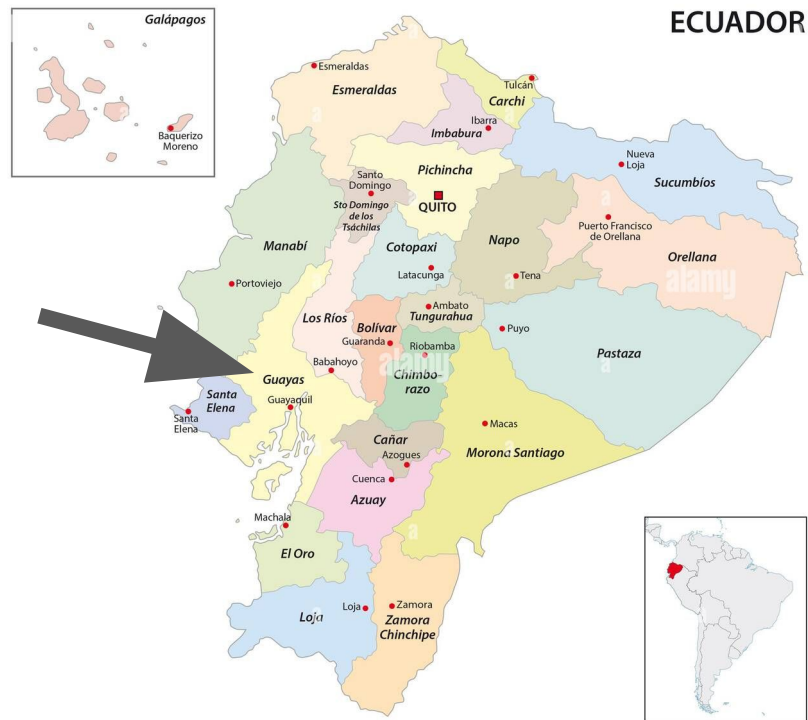
Corporación Favorita grocery sales dataset

Structure of the dataset

Daily quantity sold of a specific item in a specific store

Filtered Version used in this project

- Only the three largest product families
- Sales from January 2013 to March 2014
- Stores in the Guayas region



Project workflow

Aggregation

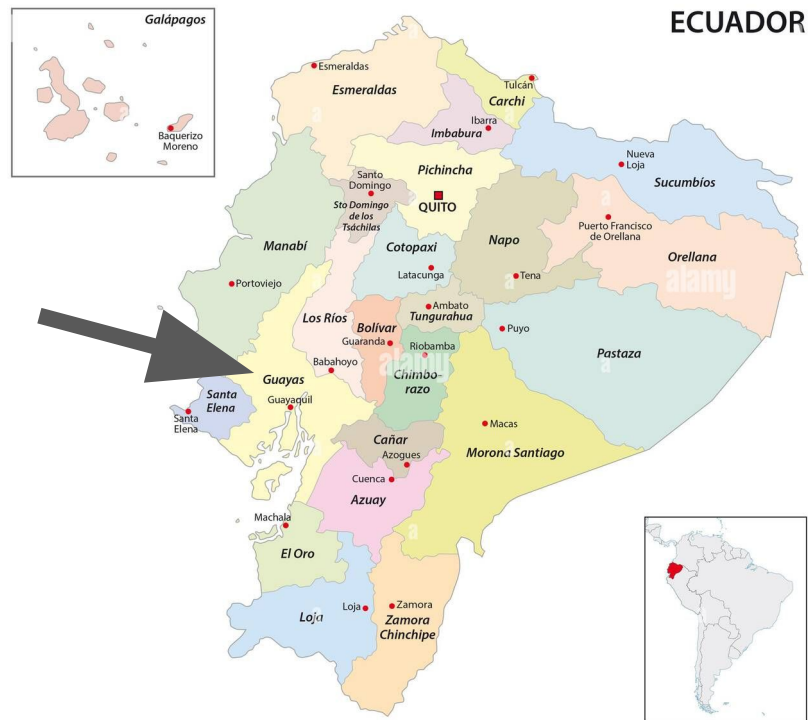
Sales aggregated by day → focusing on total daily demand

Exploratory Data Analysis

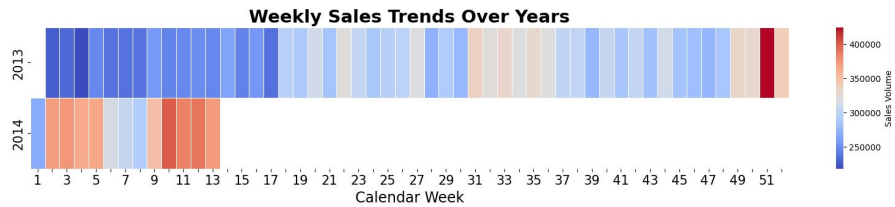
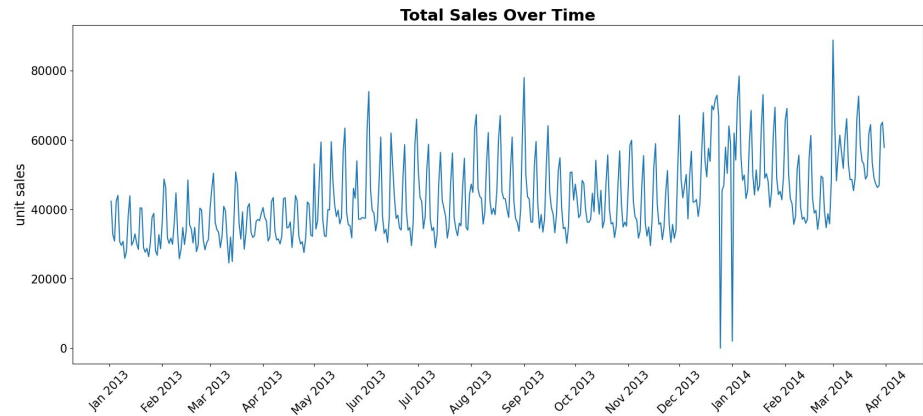
Understand patterns, trends, and weekly/seasonal effects

Model Training

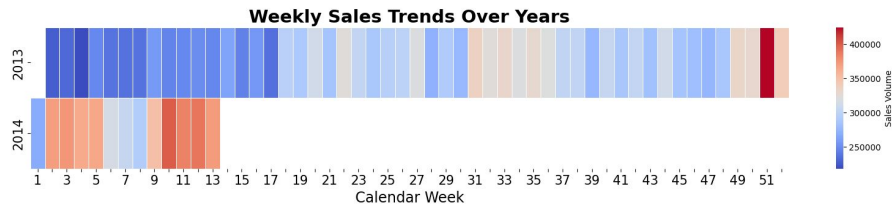
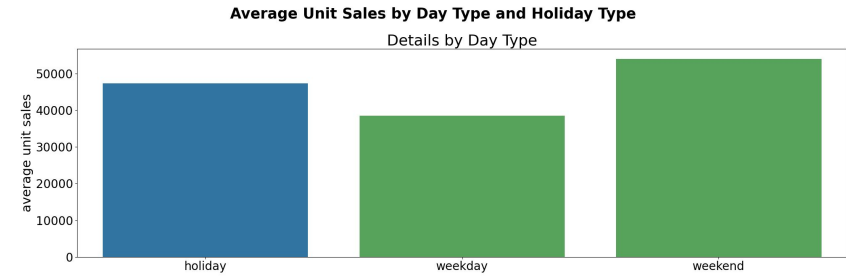
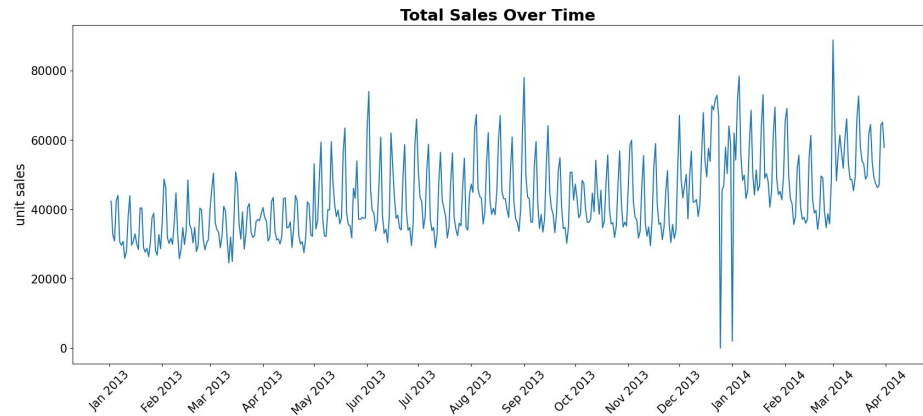
- 3 algorithms tested
- Train dataset: Jan–Dec 2013
- Test dataset: Jan–Mar 2014



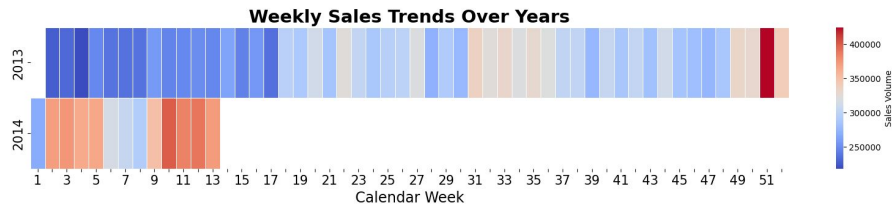
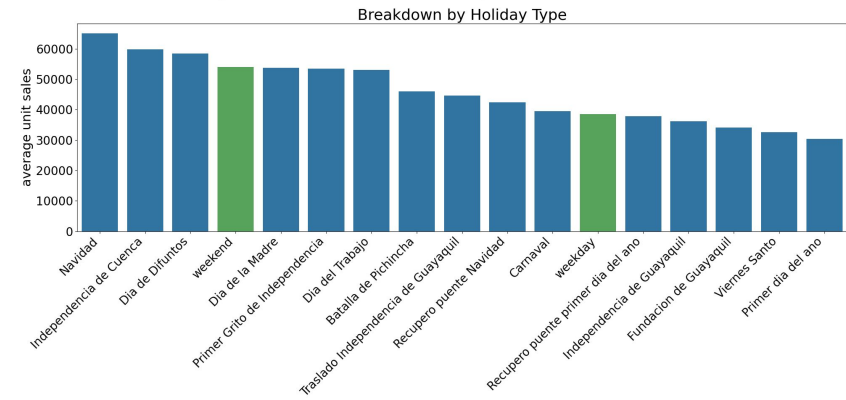
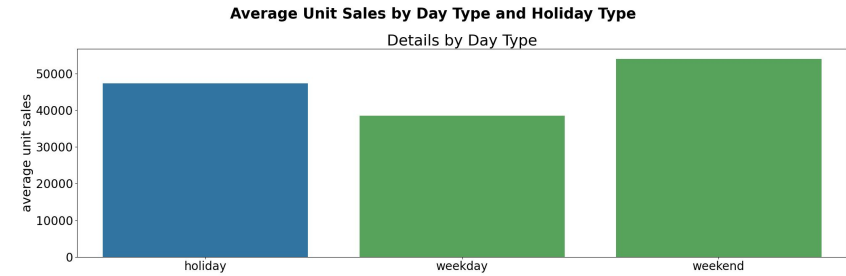
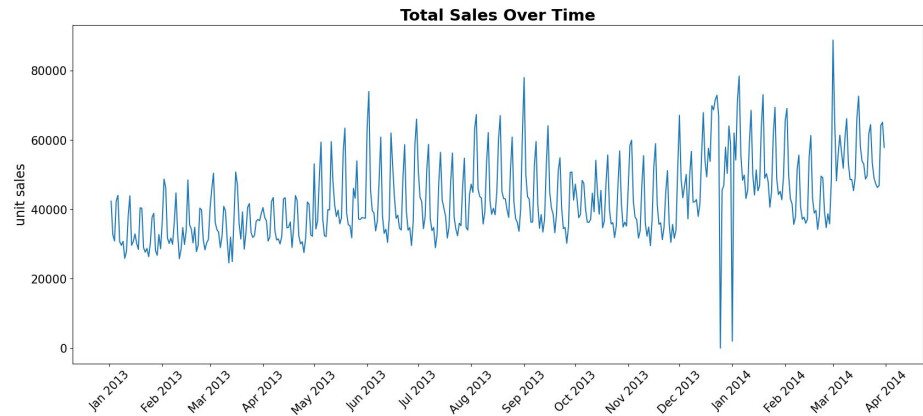
Exploratory Data Analysis



Exploratory Data Analysis



Exploratory Data Analysis



Models Tested

- Holt-Winters
- Prophet
- XGBoost

| | | |
|---------------|---|----------------|
| Train dataset | → | Jan - Dec 2013 |
| Test dataset | → | Jan - Mar 2014 |

Models Tested

- Holt-Winters
- Prophet
- XGBoost

Components

- Level
- Trend (with or without damping)
- Seasonality (single; additive or multiplicative)

Models Tested

- Holt-Winters
- Prophet
- XGBoost

Components

- Level
- Trend (with or without damping)
- Seasonality (single; additive or multiplicative)

Best Model

damped trend + additive seasonality (7 days)

Models Tested

- Holt-Winters
- Prophet
- XGBoost

Components

- Trend
- Seasonality (multiple; modeled via Fourier series)
- Holidays / Special events

Models Tested

- Holt-Winters
- Prophet
- XGBoost

Components

- Trend
- Seasonality (multiple; modeled via Fourier series)
- Holidays / Special events

Best Model

weekly + monthly seasonalities (no holidays)

Models Tested

- Holt-Winters
- Prophet
- XGBoost

Key points

- Ensemble supervised learning algorithm
- No inherent time awareness
→ manual feature engineering needed

Feature types

- Lagged values & rolling statistics
- Time-based features (day, month, day of week, ..)
- Optional exogenous features (e.g. transactions)

Recursive multi-step forecasting

predicted values used as lags for future predictions

Models Tested

- Holt-Winters
- Prophet
- XGBoost

Key points

- Ensemble supervised learning algorithm
- No inherent time awareness
→ manual feature engineering needed

Feature types

- Lagged values & rolling statistics
- Time-based features (day, month, day of week, ..)
- Optional exogenous features (e.g. transactions)

Recursive multi-step forecasting

predicted values used as lags for future predictions

Best model features

Lags: [1, 7, 30, 90, 120]

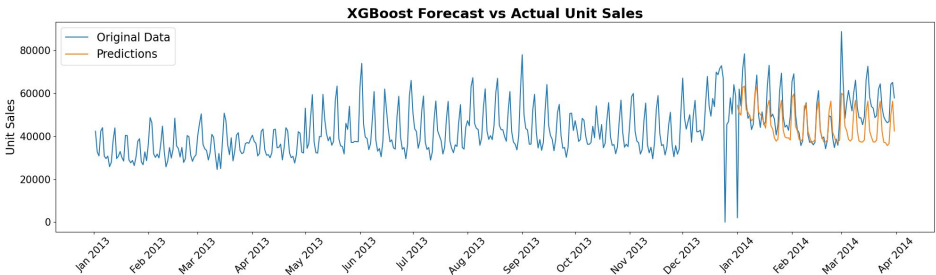
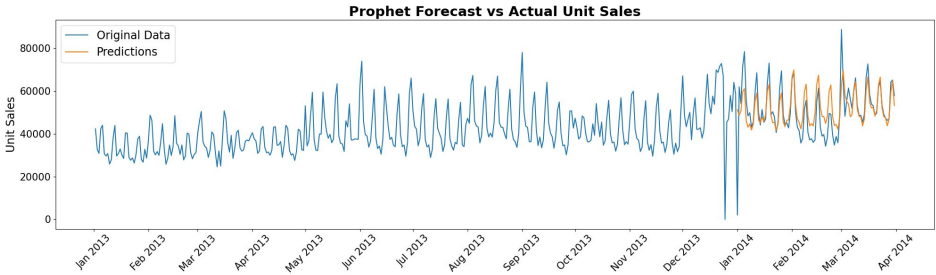
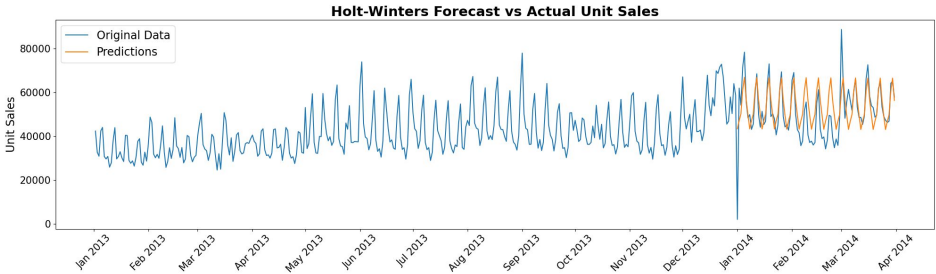
Rolling stats: mean(7), std(7)

Time-based features, cyclical encoding where relevant

No transactions

Performance Comparison

| Model | R2 | RMSE | MAE | Training Time | Prediction Time |
|--------------|--------|----------|---------|---------------|-----------------|
| Holt-Winters | 0.4333 | 9086.09 | 6509.36 | 79.12ms | 4.00ms |
| Prophet | 0.5036 | 8504.32 | 5585.90 | 208.81ms | 73.28ms |
| XGBoost | 0.2169 | 10680.99 | 7813.88 | 54.59ms | 40.94ms |



Best Model: Prophet

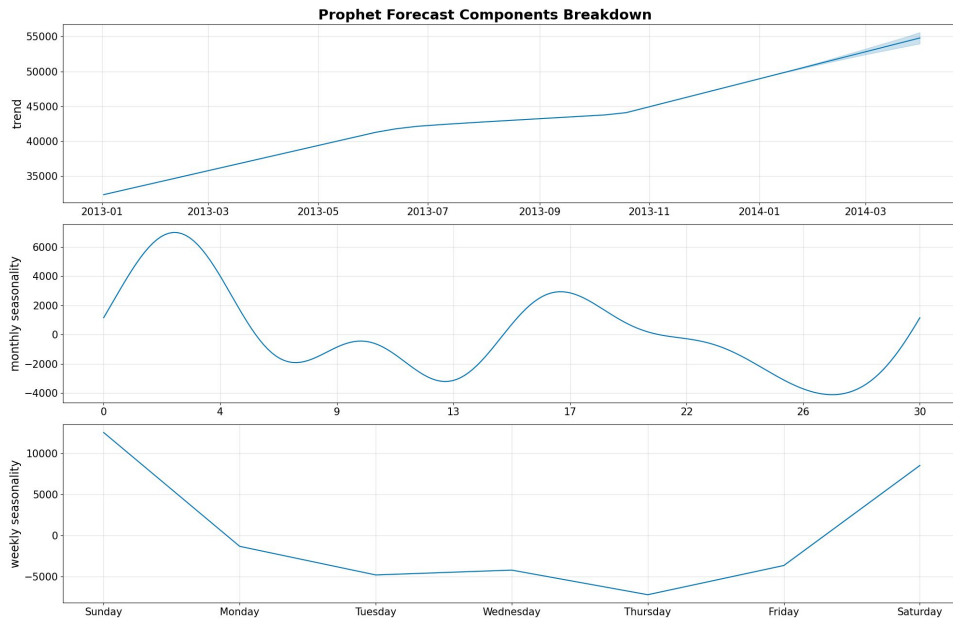
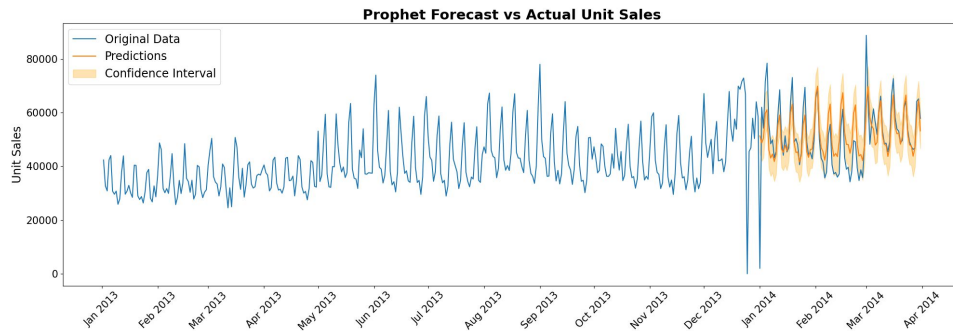
| Model | R2 | RMSE | MAE | Training Time | Prediction Time |
|--------------|--------|----------|---------|---------------|-----------------|
| Holt-Winters | 0.4333 | 9086.09 | 6509.36 | 79.12ms | 4.00ms |
| Prophet | 0.5036 | 8504.32 | 5585.90 | 208.81ms | 73.28ms |
| XGBoost | 0.2169 | 10680.99 | 7813.88 | 54.59ms | 40.94ms |

Best performance across all evaluation metrics

Time-series-specific model, simpler to use than XGBoost and more flexible than Exponential Smoothing

Rich, interpretable outputs, including components and confidence intervals

Slower in training and forecasting



Best Model: Prophet

```
# Import prophet library
from prophet import Prophet

# Initialize Prophet model with custom seasonality settings
model = Prophet(
    daily_seasonality=False,
    weekly_seasonality=False,
    yearly_seasonality=False,
    seasonality_mode='additive'
)

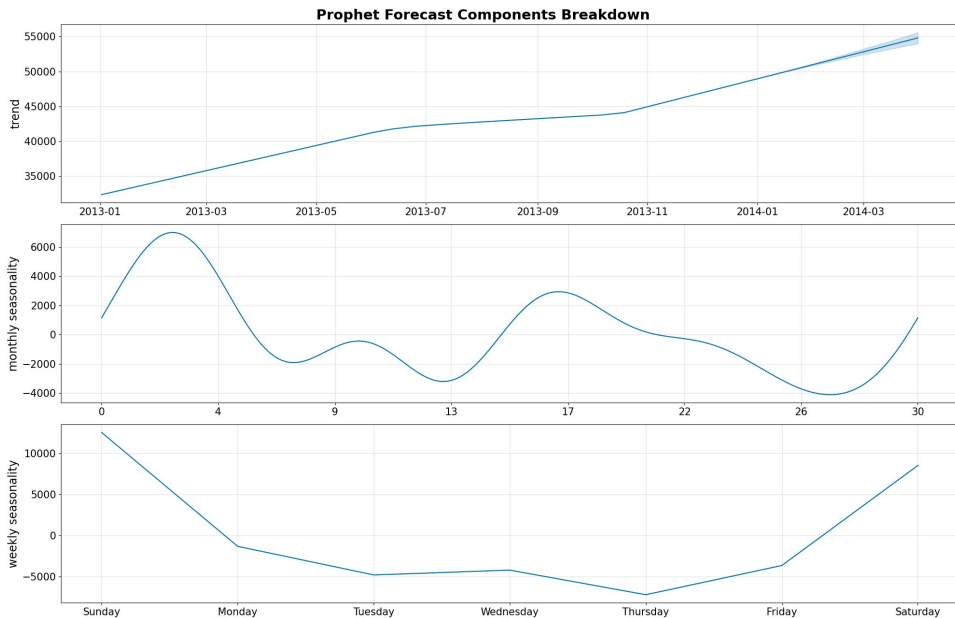
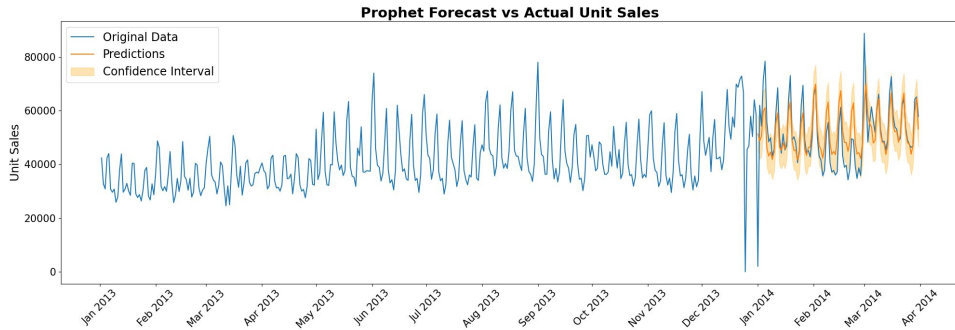
# Add custom weekly seasonality
model.add_seasonality(
    name='weekly_custom',
    period=7,
    fourier_order=5
)

# Add custom monthly seasonality
model.add_seasonality(
    name='monthly_custom',
    period=30.5,
    fourier_order=5
)

# Fit the Prophet model to the training data
model.fit(df_train)

# Create a dataframe with future dates for prediction
future_df = model.make_future_dataframe(periods=90, freq='D')

# Generate predictions
forecast = model.predict(future_df)
```



Github Repository

https://github.com/filippopedrini95/MS_DS_TimeSeriesCourse_Project.git

MS_DS_TimeSeriesCourse_Project/

- |— README.md
- |— requirements.txt
- |— data/
 - |— [CSV files]
- |— notebooks/
 - |— FirstAnalysis_&_EDA.ipynb
 - |— Model_ExponentialSmoothing.ipynb
 - |— Model_Prophet.ipynb
 - |— Model_XGBoost_skforecast.ipynb
- |— models/
 - |— [saved model files]
- |— visualizations/
 - |— [plots and charts]

Github Repository

https://github.com/filippopedrini95/MS_DS_TimeSeriesCourse_Project.git

```
MS_DS_TimeSeriesCourse_Project/  
├── README.md  
├── requirements.txt  
├── data/  
│   └── [CSV files]  
├── notebooks/  
│   ├── FirstAnalysis_&_EDA.ipynb  
│   ├── Model_ExponentialSmoothing.ipynb  
│   ├── Model_Prophet.ipynb  
│   └── Model_XGBoost_skforecast.ipynb  
├── models/  
│   └── [saved model files]  
├── visualizations/  
└── [plots and charts]
```

Thank you for your attention!