Walmart Sales Trends Predictions and Analysis via Machine Learning

Utilizing Advanced Machine Learning Techniques for Sales Insights

Umar Memon and Hamza Khan

Introduction and Objectives

Background and Motivation:

- For our project we wanted to focus on analyzing data and make predictions on it. We chose this idea because after researching about machine learning projects we wanted to do a project that would be helpful in our career and we could learn from.

Objective:

- Our main object was to analyze and make sales or demand trend predictions on retail stores.
- After researching we understood why predictions, forecasting, analyzing was important, it's for business strategies and most importantly looking at trends of product demands from a retail store.

.

Data Collection and Preprocessing

Dataset Description:

Source: Found one dataset of Walmart sales, States, Years, and other main features from data world website. We did see other Walmart dataset however this one has more features, more rows, and has data for US only. This dataset was about 300MB in size so it had a lot of data.

Key Features: Sales, unit prices, quantity order, discount, profit, and other relevant features.

Preprocessing Steps:

- Handled missing value by doing imputation techniques. Removed outliers using the IQR (interquartile range) method from statistic and probability. Used stratified random sampling to only use 1000 samples from the dataset.

- Feature Engineering, we created new features such as estimated sale using other important feature like profit, unit price, discount, sales, quantity order, any features that gave more insight in the demand of products using linear combination method. Also created log sales, log prices, and more.

Data Collection and Preprocessing

```
2 print(dataset.head())
```

```
         city  customer_age  customer_name  customer_segment  discount  \
128105  194.0          76.0          101.0               1.0      0.08
225930   77.0          65.0          475.0               1.0      0.22
355800  422.0          80.0          309.0               2.0      0.18
47215   300.0          45.0          325.0               0.0      0.22
161123   95.0          85.0          125.0               2.0      0.05

        order_date  order_id  order_priority  order_quantity  \
128105 2022-09-27     194.0             0.0            20.0
225930 2022-03-19     988.0             2.0            32.0
355800 2021-08-28     392.0             4.0             3.0
47215  2021-03-23     376.0             1.0             4.0
161123 2022-04-30     686.0             2.0            16.0

        product_base_margin  ...  profit_margin  order_date_dayofweek  \
128105                 0.49  ...      58.735697                     1
225930                 0.40  ...      21.884385                     5
355800                 0.57  ...      49.777919                     5
47215                  0.38  ...       2.756167                     1
161123                 0.59  ...       0.522490                     5

        order_date_month  order_date_year  order_date_dayssincestart  \
128105                 9             2022                       1364
225930                 3             2022                       1172
355800                 8             2021                        969
47215                  3             2021                        811
161123                 4             2022                       1214

        ship_date_dayofweek  ship_date_month ship_date_year  \
128105                    1               10           2022
225930                    2                3           2022
355800                    2                9           2021
47215                     2                3           2021
161123                    3                5           2022

        ship_date_dayssincestart  estimated_sales
128105                      1368      2355.631031
225930                      1173      2178.492221
355800                       970      1240.908999
47215                        816        19.512609
161123                      1216       150.862391

[5 rows x 38 columns]
```

```
First few rows of the dataset:
               city  customer_age       customer_name customer_segment discount  \
128105       Euless            76  Christine Abelman        Corporate     0.08
225930   Burlington            65        Sonia Cooley        Corporate     0.22
355800     Moorhead            80     Kristen Hastings      Home Office     0.18
47215         Hurst            45         Logan Currie         Consumer     0.22
161123  Chattanooga            85          Cyma Kinney      Home Office     0.05

        order_date                              order_id order_priority  \
128105  2022-09-27  34a99182-8b7a-4a61-a06d-5a61eb926f7d       Critical
225930  2022-03-19  fc94e58e-62e2-4d51-bbcf-b29469fb7e1b            Low
355800  2021-08-28  6491fc97-f7b8-4e5f-9104-1a499d586dbf  Not Specified
47215   2021-03-23  5f5daae8-77bc-47c0-a7da-bd4b4be7af8f           High
161123  2022-04-30  b2043249-8d7b-4eaa-836f-1c688a50091d            Low

       order_quantity product_base_margin  ...           product_sub_category  \
128105             20                0.49  ...             Office Furnishings
225930             32                 0.4  ...                          Paper
355800              3                0.57  ...                     Appliances
47215               4                0.38  ...                          Paper
161123             16                0.59  ...  Telephones and Communication

             profit   region     sales   ship_date         ship_mode  \
128105  23376.80793  Central       398  2022-10-04       Express Air
225930   21541.2385     East    984.32  2022-03-23       Express Air
355800  12116.94153  Central    243.42  2021-09-01       Express Air
47215     132.95754  Central     48.24  2021-03-31    Delivery Truck
161123    1052.8381    South   2015.04  2022-05-05       Regular Air

             shipping_cost      state unit_price zip_code
128105  17.759847984806232      Texas      19.98  76039.0
225930   659.3479188838185    Vermont      30.98   5401.0
355800   206.306215156242   Minnesota      81.32  56560.0
47215    19.83636696486954      Texas      12.28  76053.0
161123   710.1907635117352  Tennessee     125.99  37421.0

[5 rows x 23 columns]
```

Data Collection and Preprocessing

```python
split = StratifiedShuffleSplit(n_splits=1, test_size=n_samples, random_state=42)
for _, sample_index in split.split(filtered_df, filtered_df[stratify_column]):
    stratified_sample = filtered_df.iloc[sample_index]
```

```python
# Feature Engineering for Date Columns
for col in ['order_date', 'ship_date']:
    if col in dataset.columns:
        dataset[col + '_dayofweek'] = dataset[col].dt.dayofweek
        dataset[col + '_month'] = dataset[col].dt.month
        dataset[col + '_year'] = dataset[col].dt.year
        dataset[col + '_dayssincestart'] = (dataset[col] - dataset[col].min()).dt.days
```

```python
# Handle missing values
logging.info("Handling missing values...")  # Log message for handling missing values
imputer = SimpleImputer(strategy='mean')  # Initialize imputer to fill missing values with me
numerical_columns = dataset.select_dtypes(include=['number']).columns  # Select only numerica
dataset[numerical_columns] = imputer.fit_transform(dataset[numerical_columns])  # Apply imput

# # Normalize/Standardize data
# logging.info("Normalizing/Standardizing data...")
# numerical_features = ['customer_age', 'discount', 'order_quantity', 'product_base_margin',
scaler = StandardScaler()
# dataset[numerical_features] = scaler.fit_transform(dataset[numerical_features])

# Feature engineering: Create new features
logging.info("Creating new features...")  # Log message for feature engineering
dataset['price_adjusted_by_quantity'] = dataset['unit_price'] * dataset['order_quantity']
dataset['log_price'] = np.log1p(dataset['unit_price'])
dataset['inverse_price'] = 1 / (dataset['unit_price'] + 1e-5)
dataset['log_sales'] = np.log1p(dataset['sales'])
dataset['discount_per_quantity'] = dataset['discount'] / (dataset['order_quantity'] + 1e-5)
dataset['profit_margin'] = dataset['profit'] / (dataset['sales'] + 1e-5)
```

```python
# Calculate estimated sales based on the features
intercept = 0.1  # Placeholder value
coef_quantity = 0.4  # Placeholder value
coef_price = 0.3  # Placeholder value
coef_discount = 0.2  # Placeholder value
coef_profit = 0.1
coef_log_sales = 0.1
coef_discount_per_quantity = 0.05
coef_profit_margin = 0.05
coef_log_price = 0.1

dataset = dataset[dataset['order_date_year'] != 2023]

dataset['estimated_sales'] = abs(
    intercept
    + coef_quantity * dataset['order_quantity']
    + coef_price * dataset['unit_price']
    + coef_discount * dataset['discount']
    + coef_profit * dataset['profit']
    + coef_log_sales * dataset['log_sales']
    + coef_discount_per_quantity * dataset['discount_per_quantity']
    + coef_profit_margin * dataset['profit_margin']
    + coef_log_price * dataset['log_price']
)
```

```python
# Function to check for negative values
def check_for_negative_values(dataset, stage):
    logging.info(f"Checking for negative values after {stage}...")  # Log message for checking negat
    numerical_features = dataset.select_dtypes(include=[np.number]).columns  # Get numerical columns
    for feature in numerical_features:  # Iterate through numerical columns
        if (dataset[feature] < 0).any():  # Check if there are negative values
            dataset[feature] = abs(dataset[feature])  # Convert negative values to positive
            logging.warning(f'Negative values found and converted to positive in {feature} after {st
```

(EDA) Exploratory Data Analysis

Data Exploration:

Libraries: Imported various libraries, like pandas, numpy, seaborn, matplotlib, linear regression models, time series, and others for forecasting and visualizations.

Data Cleaning: Removed duplicates, corrected data types, and dropped rows/columns with missing data.

Data Transformation: Performed data transformation converting date column to datetime format for time series analysis.

Descriptive Statistics: Calculated statistics for numerical column, providing insight in dispersion, central tendency, and shape of the data distribution.

Correlation Analysis and Distribution Visualization: computed correlation matrix and visualized it using heatmap to see relationships between different features. Visualization distributions and relationships of numerical variables using box plots, bar graphs, scatter plots, and histograms.

Exploratory Data Analysis

```python
from sklearn.preprocessing import PolynomialFeatures
import seaborn as sns
import matplotlib.pyplot as plt
import logging
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

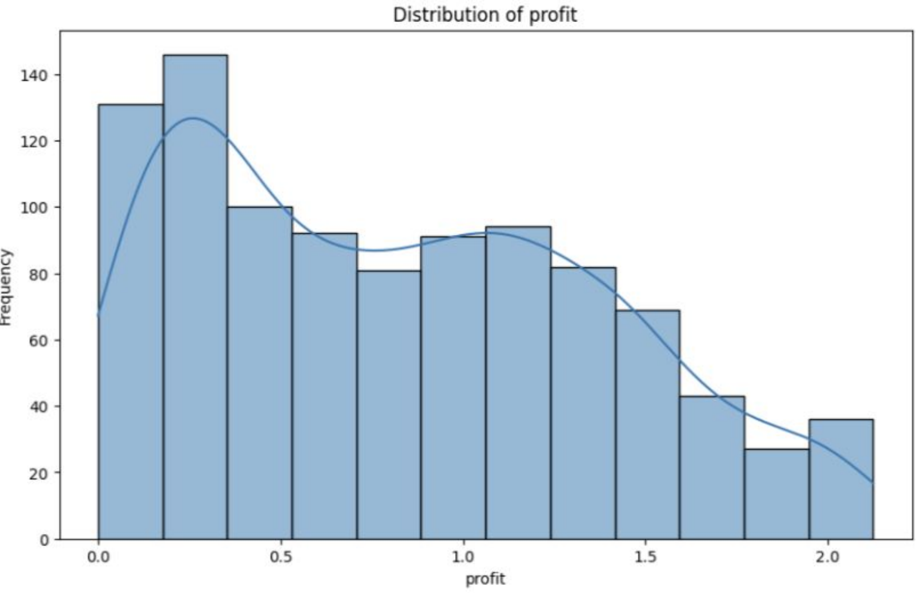
```python
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.model_selection import train_test_split
```

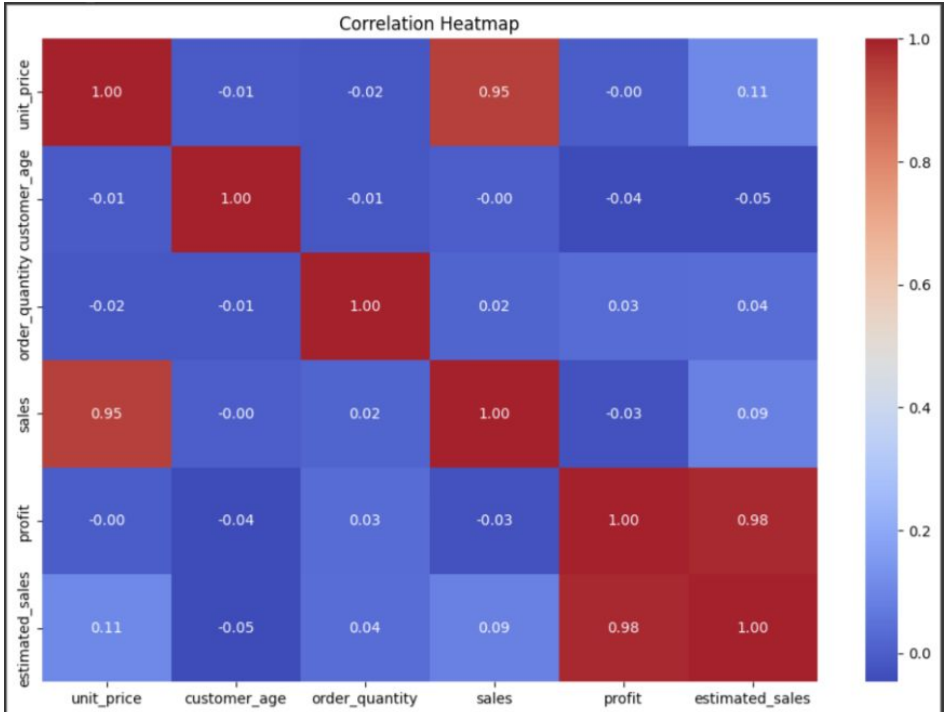
```python
from sklearn.linear_model import Lasso, LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler, RobustScaler, MinMaxScaler, MaxAbsScaler
from sklearn.model_selection import RandomizedSearchCV, train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import logging
import numpy as np
```

```python
# Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedShuffleSplit
import logging
import pandas as pd
import numpy as np
from google.colab import files, drive
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
import logging
import matplotlib.pyplot as plt
import seaborn as sns
```

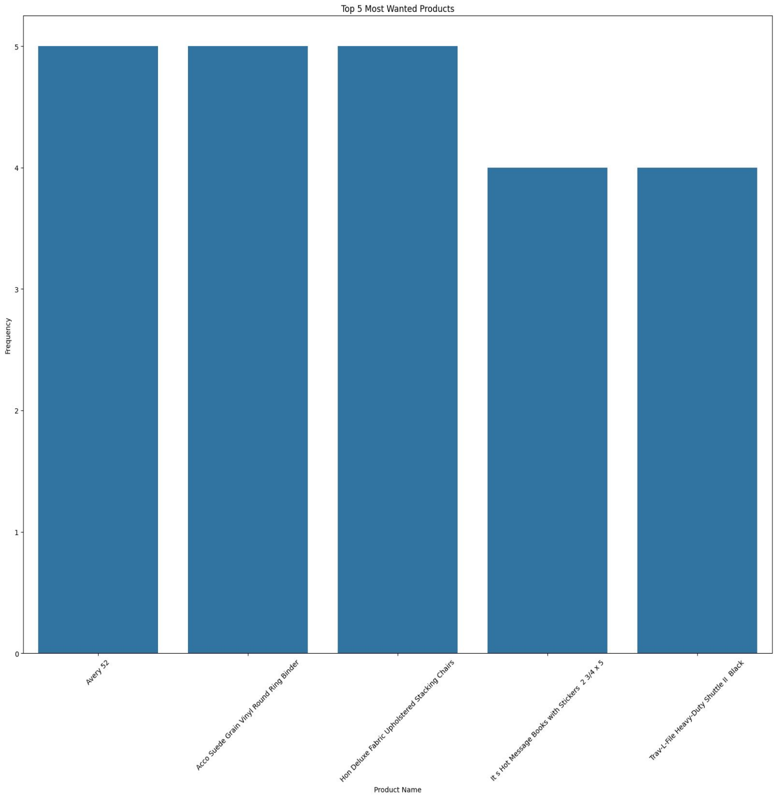
```python
from prophet import Prophet
from prophet.plot import add_changepoints_t
```

Exploratory Data Analysis (Demo)





Exploratory Data Analysis (Demo)



Exploratory Data Analysis

```python
from sklearn.preprocessing import PolynomialFeatures, LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
import logging
import numpy as np
import pandas as pd

# Function to calculate correlation
def calculate_correlation(data):
    correlation_price_sales = data[['unit_price', 'estimated_sales']].corr()
    logging.info(f"Correlation between Unit Price and Estimated Sales:\n{correlation_price_sales}")

    correlation_sales_estimated = data[['sales', 'estimated_sales']].corr()
    logging.info(f"Correlation between Sales and Estimated Sales:\n{correlation_sales_estimated}")

    return correlation_price_sales, correlation_sales_estimated

# Function to visualize the relationship between features and estimated sales
def visualize_relationship(data):
    plt.figure(figsize=(10, 6))
    plot_df = pd.DataFrame({
        'unit_price': data['unit_price'].values.ravel(),
        'estimated_sales': data['estimated_sales'].values.ravel()
    })
    sns.scatterplot(x='unit_price', y='estimated_sales', data=plot_df)
    plt.title('Relationship between Unit Price and Estimated Sales')
    plt.xlabel('Unit Price')
    plt.ylabel('Estimated Sales')
    plt.show()
```

```python
# Function to visualize the relationship between features and estimated sales
def visualize_relationship(data):

    features = ['unit_price', 'order_quantity', 'discount', 'profit', 'sales']
    for feature in features:
        if feature in data.columns and 'estimated_sales' in data.columns:  # Check if columns exi
            plt.figure(figsize=(10, 6))
            plot_df = pd.DataFrame({
                feature: data[feature].values.ravel(),  # Ravel the array to ensure 1D
                'estimated_sales': data['estimated_sales'].values.ravel()  # Ravel the array to e
            })
            sns.scatterplot(x=feature, y='estimated_sales', data=plot_df)  # Pass DataFrame to sc
            plt.title(f'Relationship between {feature} and Estimated Sales')
            plt.xlabel(feature)
            plt.ylabel('Estimated Sales')
            plt.show()
        else:
            logging.warning(f"Skipping visualization for {feature} due to missing column(s).")

#Ravel flattens a multi dimensiol array
```

Models

Supervised Learning:

Regression Models: Linear Regression, Lasso Regression, Ridge Regression, Elastic, Random Forest, Gradient Boosting, Neural Network, and Ensemble model (combining all of these model for better performance).

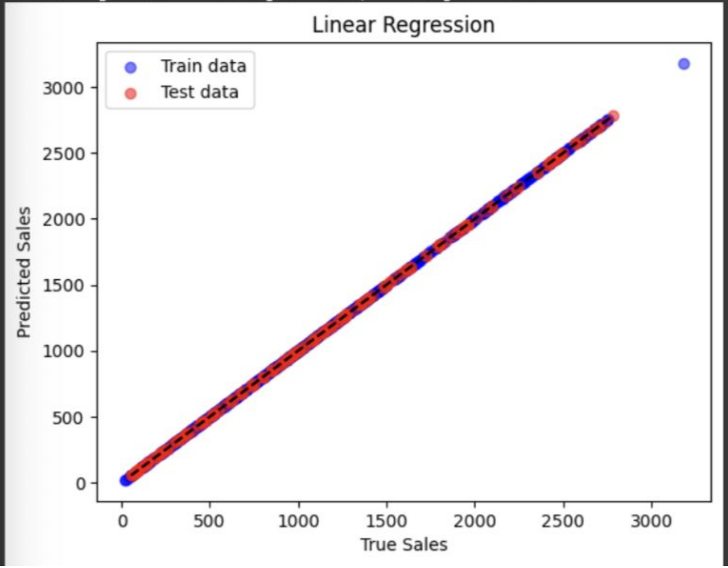
Training and Validation: Performed training and testing split, hyperparameter tuning using Grid and Randomized Search to enhanced each models parameters. Cross-Validation techniques to ensure robust performance evaluations. Used pipeline, streamlining a machine learning process steps and param-grid defining set of hyperparameters tuning the models. Also did standardizing/Normalizing on models which had benefits.

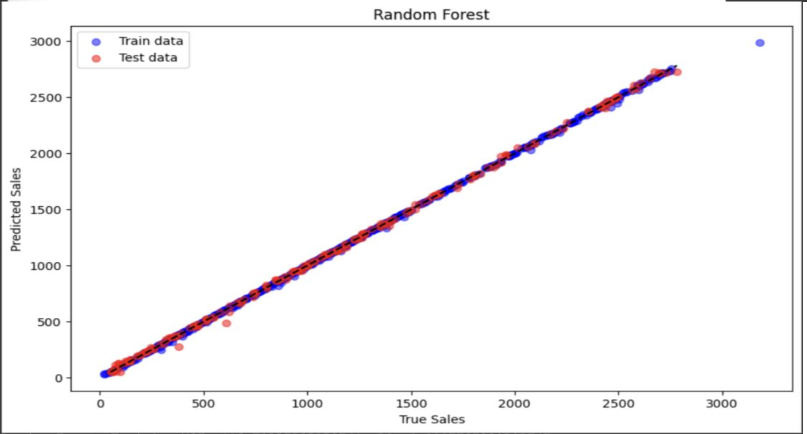
Performance of Regression Model and Forecasting: Used R2 statistical measure to evaluate how well a regression model fit data. Also used MSE (Mean Squared Error) to evaluate the performance of predictive models. The closer the R2 is to 1 means the model is performing well, and if MSE has lower values then the model is performing well. Used Ensemble Model to forecast.

Time Series Forecasting Models: ARIMA, LSTM, XGBoosting, and Prophet. Used to forecast estimated sales for future (2025-2026). ARIMA (Autoregressive Integrated Moving Average) uses past values/errors to predict future values. LSTM (Long Short Term Memory) captured patterns in time series data and its a Recurrent Neural Network. XGBoost (Extreme Gradient Boosting) creates lag feature, rolling statistic and time related features. Prophet was developed for time series forecasting by Facebook.

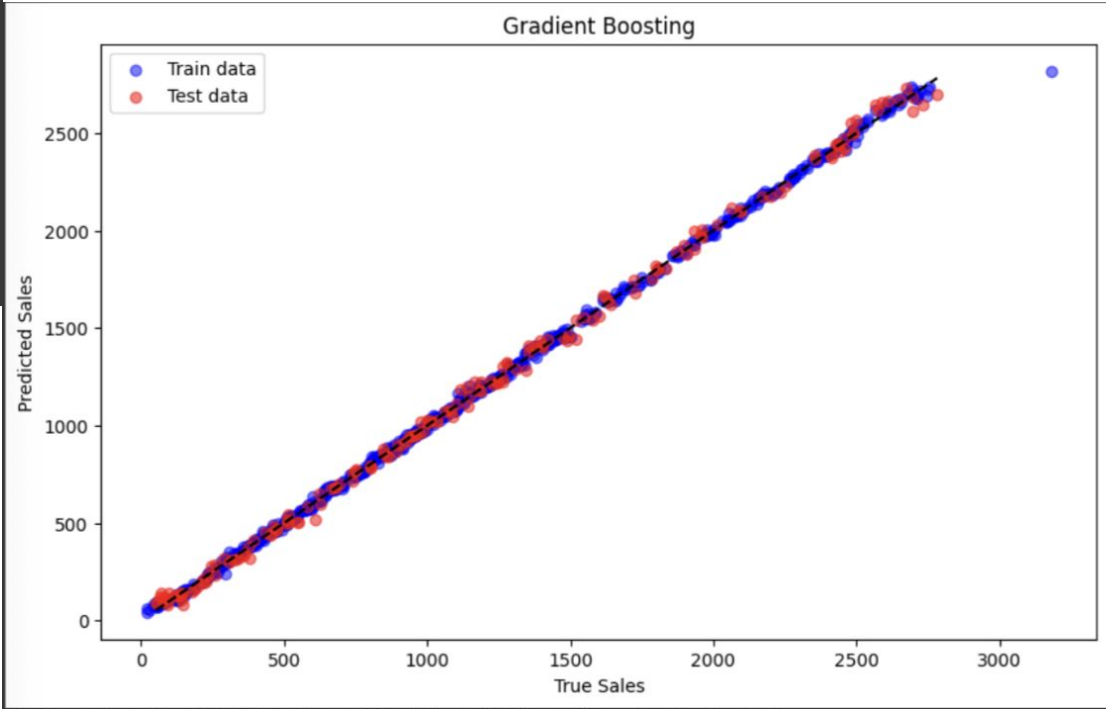
Demo of Models

```
Linear Regression Training MSE: 1.2154432126440646e-24, Testing MSE: 1.1103425
Linear Regression Training R²: 1.0, Testing R²: 1.0
```



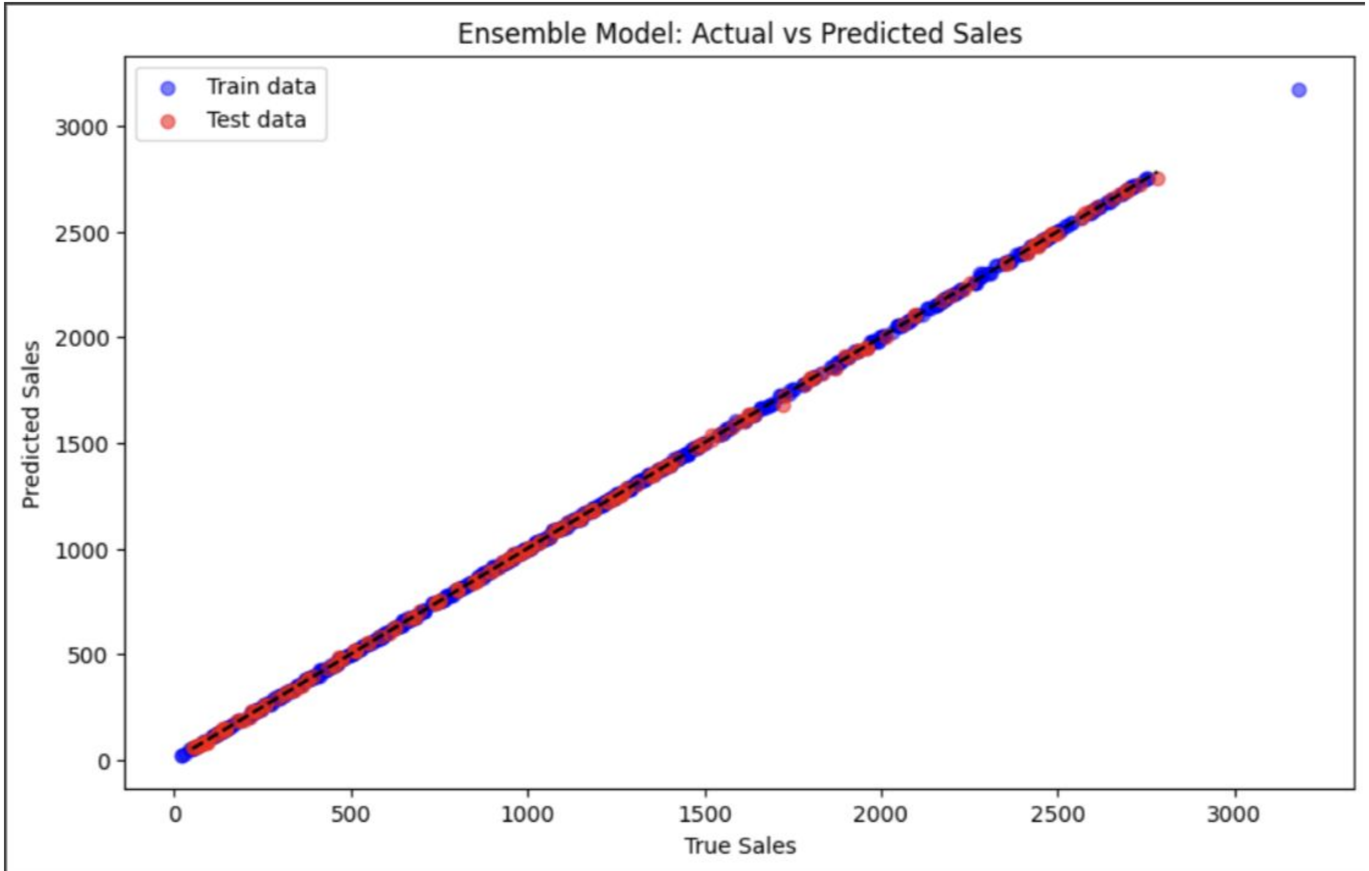



```
Random Forest Training MSE: 116.79235285359741, R²: 0.9997785220303747
Random Forest Testing MSE: 422.1751271508969, R²: 0.9993103597859397
Random Forest Cross-validated MSE: 968.32 (+/- 986.96)
```



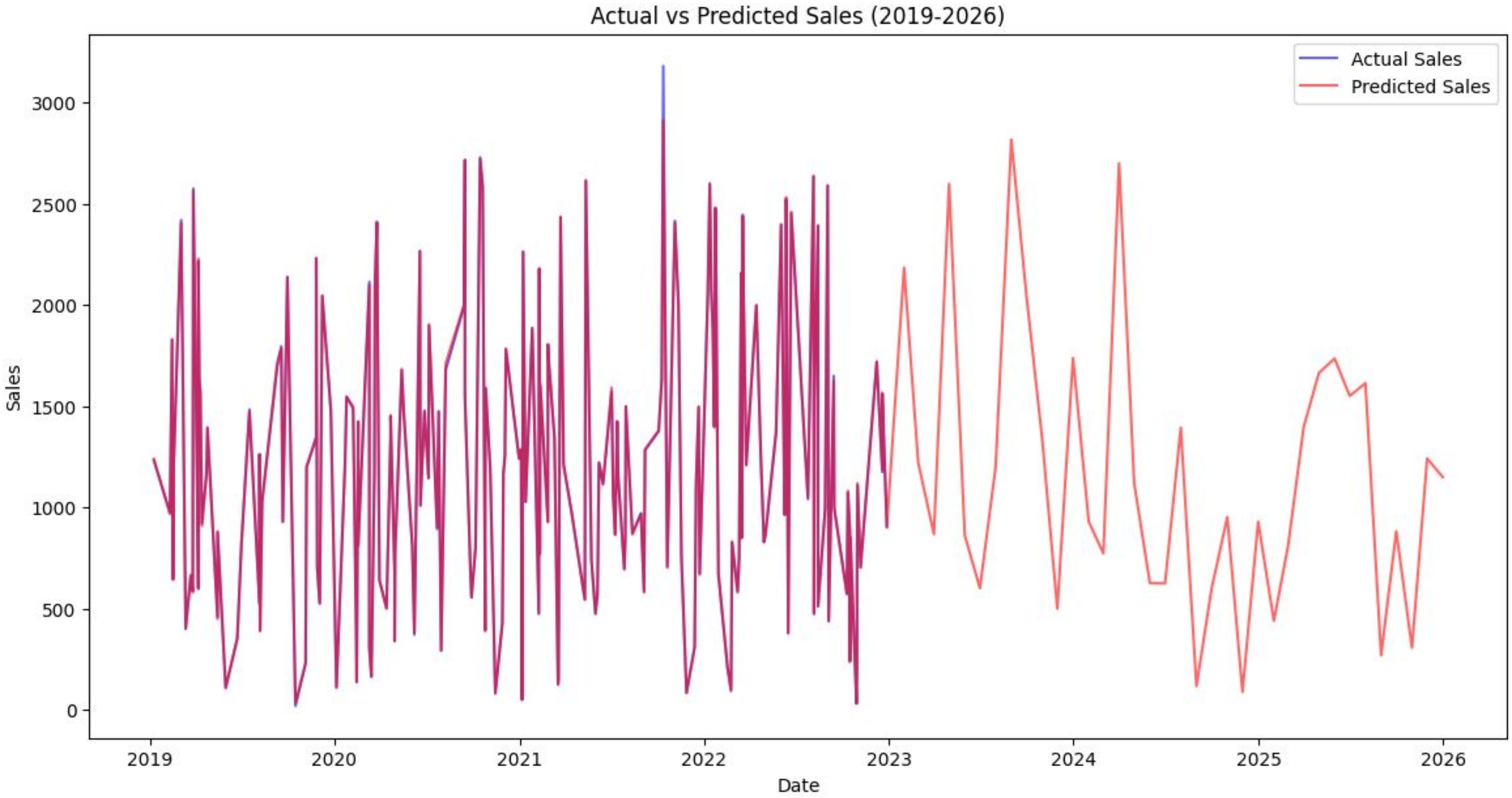

```
Gradient Boosting Training MSE: 423.11296808273687, R²: 0.999197634101861
Gradient Boosting Testing MSE: 1099.1822946247576, R²: 0.998204441085689
Gradient Boosting Cross-validated MSE: 1500.59 (+/- 696.61)
```

Demo of Models

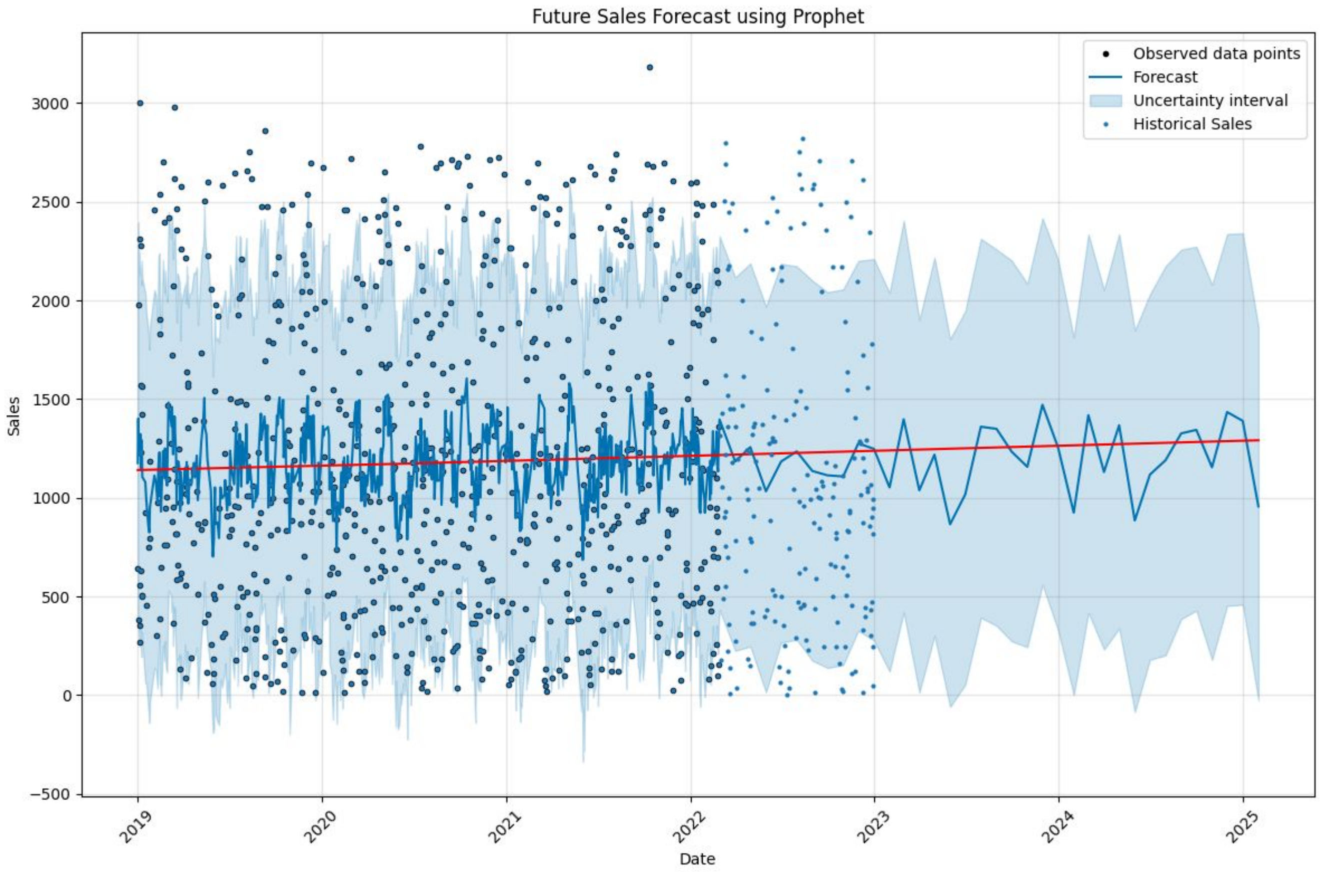



```
Ensemble Model Training MSE: 20.984008550257567, R²: 0.9999602071925536
Ensemble Model Testing MSE: 54.31190110669228, R²: 0.9999112793040225
```

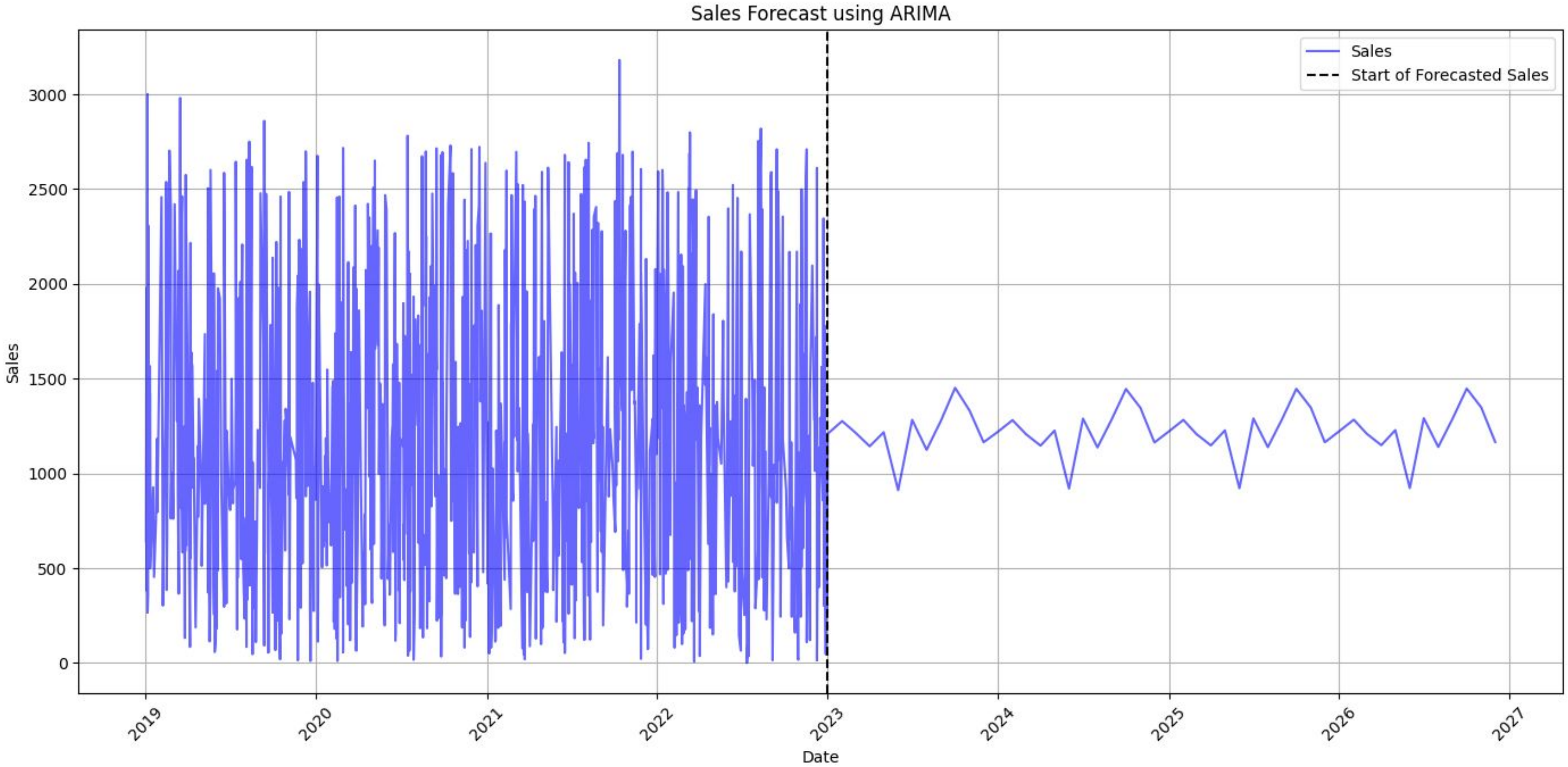
(Demo) Forecasting with Regression Ensemble Model



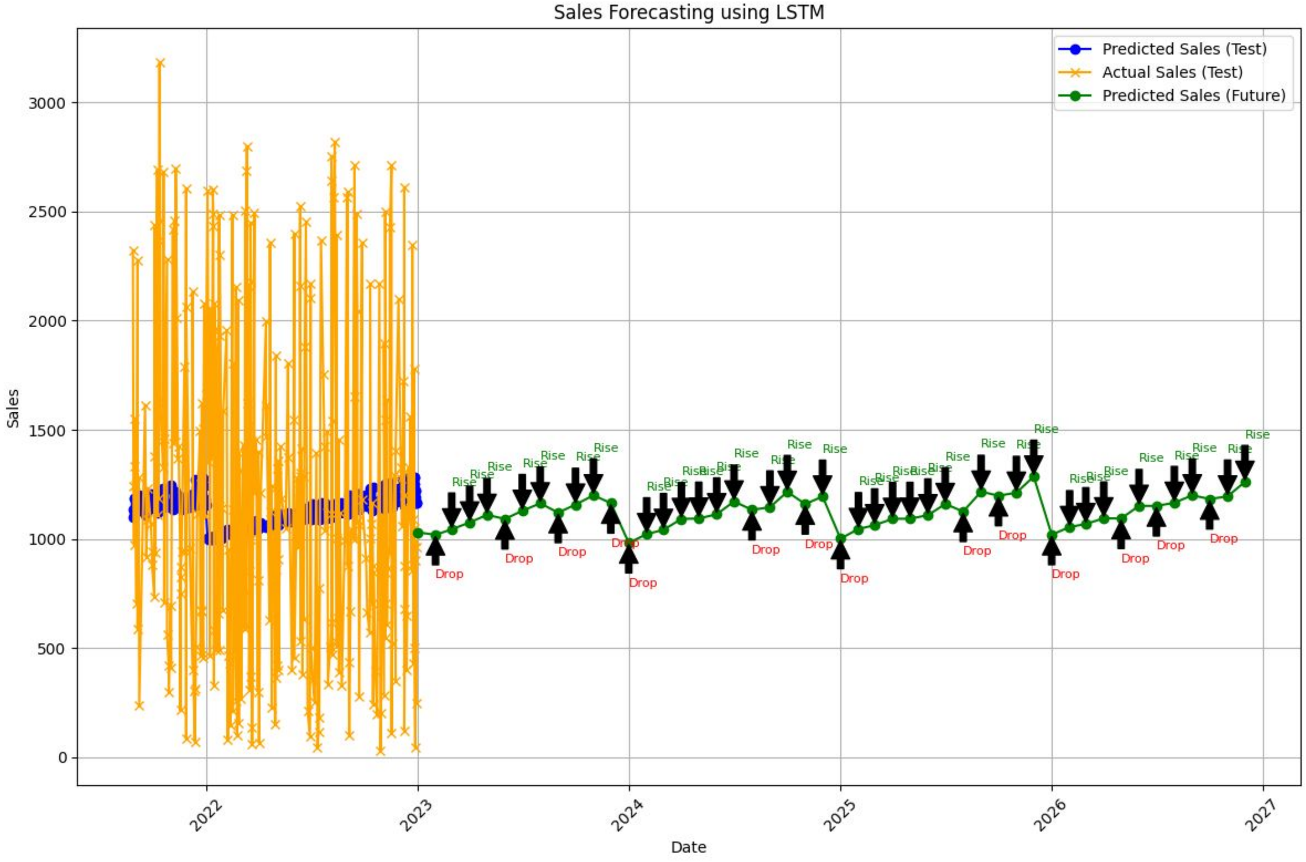
(Demo) Forecasting with Time Series Model (Best Model)



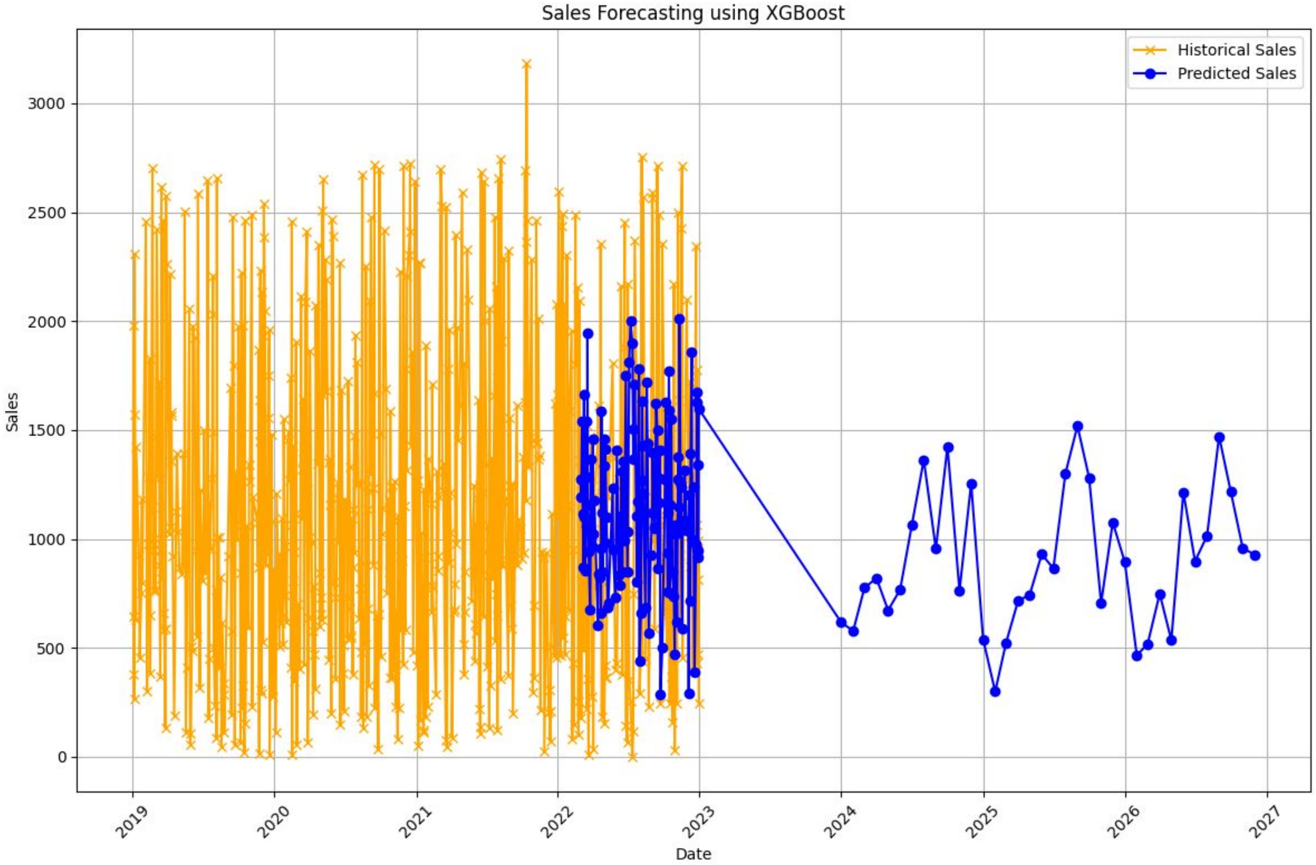
(Demo) Forecasting with Time Series Model



(Demo) Forecasting with Time Series Model



(Demo) Forecasting with Time Series Model



Models (Code Snippets)

```python
# fit the Prophet model with holidays and additional regressors
model = Prophet(holidays=holidays, yearly_seasonality=True, weekly_
# Add custom seasonalities for more depth and interpertations
model.add_seasonality(name='monthly', period=30.5, fourier_order=5

# Add regressors (if any)
# model.add_regressor('additional_feature')

p_model = model.fit(train_data)

# Make predictions on the test set
future = model.make_future_dataframe(periods=len(test_data), freq=
forecast = model.predict(future)

# Plot the results
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)
fig = model.plot(forecast, ax=ax)
add_changepoints_to_plot(fig.gca(), model, forecast)
plt.plot(dataset['ds'], dataset['y'], 'o', markersize=2, label='Hi
plt.legend()
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales Forecasting using Prophet')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()

# Generate future dates for 2024, 2025, and 2026
future_dates = model.make_future_dataframe(periods=3*12, freq='M')
future_forecast = model.predict(future_dates)
```

```python
def gradient_boosting(X_train, y_train, X_test, y_test):


    # Convert datetime columns to numeric (Unix timestamp) before scaling
    # for col in X_train.select_dtypes(include=['datetime64']).columns:
    #     X_train[col] = X_train[col].astype(np.int64) // 10**9
    #     X_test[col] = X_test[col].astype(np.int64) // 10**9


    pipeline = Pipeline([
        ('scaler', StandardScaler()),  # Placeholder scaler, will be set in GridSearch(
        ('poly', PolynomialFeatures()),  # Add polynomial features
        ('regressor', GradientBoostingRegressor(random_state=42))
    ])

    param_grid = {
        'scaler': [StandardScaler(), RobustScaler(), MinMaxScaler(), MaxAbsScaler()],
        'poly__degree': [1],  # Polynomial degree
        'regressor__n_estimators': [100,200,300,400,500,600,700,800,900,1000,2000,3000,
        'regressor__learning_rate': [0.000001,0.000000000001,0.001, 0.0001,0.01, 0.05,
        'regressor__max_depth': [1,2,3,4,5,6,7,8,9,10,20,30, 40,60,70,80,90,100],  # Ma
        'regressor__min_samples_split': [2, 5,7,9,10,12,14,20,30,40,45,50,55,60,65,70,7
        'regressor__min_samples_leaf': [1, 2,8,9,10,12,15,20,30,40,50,55,65,70,75,80,90
        'regressor__subsample': [0.0001,0.1,0.2,.3,.4,.5,.6,0.7, 0.8, 0.9, 1.0],  # Fra
    }

    # Hyperparameter tuning with GridSearchCV
    # search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_error

    # search = RandomizedSearchCV(pipeline, param_grid, n_iter=50, cv=5, scoring='neg_m
    # search.fit(X_train, y_train)

    # # Get the best model from the hyperparameter search
    # best_model = search.best_estimator_
    # print(f"Best parameters for Gradient Boosting Regressor: {search.best_params_}")

    # Perform cross-validation and get the mean score
    # cv_scores = cross_val_score(best_model, X_train, y_train, cv=5, scoring='neg_mean

    search = RandomizedSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_e
    search.fit(X_train, y_train)
    best_model = search.best_estimator_

    # Train the best model on the entire training set
    best_model.fit(X_train, y_train)

    # Make predictions on the training and testing sets
    train_predictions = best_model.predict(X_train)
    test_predictions = best_model.predict(X_test)
```

```python
# Ensemble function
def ensemble_model30(X_train, y_train, X_test, y_test):
    """
    Function to create and evaluate an ensemble model using stacking of multiple regression m
    """

    # Convert datetime columns to numeric (Unix timestamp)
    for col in X_train.select_dtypes(include=['datetime64']).columns:
        X_train[col] = X_train[col].astype(np.int64) // 10**9
        X_test[col] = X_test[col].astype(np.int64) // 10**9

    # Define base models
    base_models = [
        ('random_forest', random_forest(X_train, y_train, X_test, y_test)[0]),
         ('gradient_boosting', gradient_boosting(X_train, y_train, X_test, y_test)[0]),
        ('decision_tree', decision_tree(X_train, y_train, X_test, y_test)[0]),
        ('neural_network', neural_network(X_train, y_train, X_test, y_test)[0]),
        ('linear_regression', linear_regression_model(X_train, y_train, X_test, y_test)[0]),
        ('elastic_net', elastic_net(X_train, y_train, X_test, y_test)[0]),
        ('lasso_regression', lasso_regression_model(X_train, y_train, X_test, y_test)[0]),
        ('ridge_regression', ridge_regression(X_train, y_train, X_test, y_test)[0])

    ]

    # Define stacking regressor
    ensemble = StackingRegressor(
        estimators=base_models,
        final_estimator=GradientBoostingRegressor(),  # Final estimator
        passthrough=True
    )

    # Train the ensemble model
    logging.info("Training the ensemble model...")
    ensemble.fit(X_train, y_train)

    # Make predictions
    train_predictions = ensemble.predict(X_train)
    test_predictions = ensemble.predict(X_test)
```

Conclusion

Results and Limitations:

- The accuracy of the forecasting of Walmart sales for the future depends on many factors. Highest estimated sales were in the range of 0-3000$, most products were home appliances excluding other products sold at Walmart. Used 1000 samples due to execution time for models. Weren't able to find exact real trends specifically for US, graphs, and other specifics to compare with our findings.
- Researched and coded using various websites such as stackoverflow, geeksforgeeks, and others however we did struggle to find specific tutorials on complex coding areas. Used Google Colab but were limited on which GPU to use.

Key Findings:

- Linear Regression, Ensemble, and Prophet models, were the only three that gave us the best in fitting on data and predicting. The two regression models had better R2 and MSE. These two models can do good on unseen data. We were surprised LR did a bit better than Random Forest and Gradient Boosting, we were expecting RF and GB to do better after researching about it on complex data.
- Prophet was the only model out of the other 3 time series model that gave us a better and accurate results. We spent a lot of time trying to work with Arima, LSTM, XGBoosting but these three gave us more confusing forecasting which look off from the historical sales. But Prophet gave us a easier interpretations and better representation.
- Ensemble and Prophet model do have some similarities only in the trends of sales.

Conclusion

Model	Training MSE	Testing MSE	Training R^2	Testing R^2
Lasso Regression	0.4399621278	3.3189848872	0.9999992245	0.9999933602
Random Forest	143.9248826342	4838.8292807894	0.9997463171	0.9903197451
Gradient Boosting Model	5574.9082003059	4134.8262450875	0.9901736321	0.9917281289
Decision Tree	0.8602539747	3586.6020339802	0.9999984837	0.9928248715
Neural Network	951.6560332385	11892.8469081357	0.9981953369	0.9805725516
Linear Regression	1.2154432126	1.110342550026471	1.0000000000	1.0000000000
Elastic Net	4.6861899849	2.6315806283	0.9999911134	0.9999957012
Ridge Regression	1.1381866	1.388515542	1.0000000000	1.0000000000
Ensemble Model	20.9840085503	54.3119011067	0.9999602072	0.9999112793

Model	Training MSE	Testing MSE	Training R^2	Testing R^2
Lasso Regression	0.4399621278	3.3189848872	0.9999992245	0.9999933602
Random Forest	143.9248826342	4838.8292807894	0.9997463171	0.9903197451
Gradient Boosting Model	5574.9082003059	4134.8262450875	0.9901736321	0.9917281289
Decision Tree	0.8602539747	3586.6020339802	0.9999984837	0.9928248715
Neural Network	2345.0347563957	10289.5367098171	0.9958666272	0.9794154056
Linear Regression	5.951888742	7.3544236781	1.0000000000	1.0000000000
Elastic Net	155.8785344817	203.1706101532	0.9997252475	0.9995935498
Ridge Regression	1.17254906	2.389266	1.0000000000	1.0000000000
Ensemble Model	20.5206590949	343.5874968719	0.9999638302	0.9993126406

Conclusion

Mistakes:

- We made mistakes during this project. Spent too much finding the right dataset, didn't research enough on specific areas until later, experimented too much, changed our goal many times due to limited dataset causing a redo on the project, confused on certain steps, specified or generalized the goal, didn't use the best IDE, and took time to realize how to decrease execution times so we could adjust the code for better results.

What we learned:

- We learned to first have a better plan or outline of the project, to stick with it, use better IDE, and do more research before coding. To write code in blocks step by step, better for readability, and commenting. Learned alot about the models, how to use these tools from the libraries, and how to research. Understood concepts of machine learning.

What's for the future?:

- For future work we plan to make our goals more complex in this area of predictions and analysis of data and working with it. To focus more on these models, explore advanced models, and expand on this project to make improvements for career learning.

Q&A

Thank you for your attention !!!

Why did the predictive model apply for a job? It wanted to improve its training.

And why was the predictive model always relaxed? Because it never overfitted.

But why was the predictive model always invited to the strategy meetings? Because it could see the future!

Questions?