Predicting Monthly Product Demand Using Linear Regression

Submitted by:
Group Members

1. Abdellah Teshome - ATE/0407/13
2. Biniyam Dagne - ATE/1540/13
3. Abdurezak Zeynu - ATE/7317/13

Course: ML & BD
Instructor: Mr. Bisrat Bekele
Submission Date: 12\25\2024

1. Introduction

Context

Efficient warehouse management is critical for businesses to minimize operational costs, reduce waste, and meet customer demands. Predicting product demand accurately is one of the most important aspects of inventory management. Effective demand forecasting helps avoid issues such as overstocking, which leads to increased holding costs, or stockouts, which result in lost sales and unsatisfied customers.

Problem Statement

Forecasting product demand is challenging due to several factors, including seasonal variations, varying demand trends across product categories, and changes in consumer behavior. These complexities necessitate a reliable and efficient model to predict demand and support better inventory management decisions.

Objective

This project aims to develop a linear regression model that predicts monthly product demand based on:

- Historical sales data,
- Seasonal factors (month of the year), and
- Product categories.

2. Dataset Overview

Dataset Description

The dataset used for this project is a synthetic dataset generated specifically to simulate monthly product demand in a warehouse. It contains the following features:

Feature	Description	Type
Month	The month of the year (1 to 12).	Numerical
Product_Category	The category of the product (e.g., Confectionery, Beverages, Snacks).	Categorical
Previous_Sales	Sales volume of the product in the previous month.	Numerical
Seasonality_Factor	A factor representing seasonal variations for demand.	Numerical
Demand	The target variable, representing actual product demand.	Numerical

Dataset Source

The dataset was generated synthetically using a Python script. It includes seasonal trends, categorical product information, and random noise to simulate real-world variability.

Dataset Preparation

1. **Handling Missing Values**:
    - Checked for and handled any missing data to ensure data integrity.
    - Result: No missing values were found.
2. **One-Hot Encoding**:
    - The `Product_Category` feature was encoded using one-hot encoding, creating binary columns for each category.
3. **Data Splitting**:
    - The dataset was split into training (80%) and test (20%) sets to evaluate the model's performance on unseen data.

3. Exploratory Data Analysis (EDA)

Statistical Insights

- Summary statistics for numerical features:
  - **Demand**: Ranges from 100 to 900, with a median around 400–600.
  - **Previous Sales**: Ranges from 50 to 500, showing variability across product categories.
- **Correlation**: Strong correlation observed between `Previous_Sales` and `Demand` (approximately 0.85).

Visualizations

1. **Histogram of `Demand`**:
   - Distribution shows most demand values are between 300 and 700.
2. **Scatter Plot: `Previous_Sales` vs. `Demand`**:
   - Reveals a strong positive correlation.
3. **Box Plot of `Demand` by `Product_Category`**:
   - `Confectionery` has the highest median demand, while `Beverages` show more variability.
4. **Correlation Heatmap**:
   - Highlights strong correlations between numerical features like `Previous_Sales` and `Demand`.

4. Data Preprocessing

Steps Taken

1. **Handling Missing Values**:
   - The dataset was checked for missing values to ensure data integrity.
   - Result: No missing values were found in the dataset.
2. **Feature Encoding**:
   - The `Product_Category` feature was encoded using **one-hot encoding** to transform categorical data into numerical form. This created separate binary columns for each category (e.g., `Confectionery`, `Beverages`, and `Snacks`).
3. **Splitting the Dataset**:
   - The dataset was split into **training (80%)** and **testing (20%)** sets to evaluate the model's performance on unseen data.

Code Snippet

[24]:
```python
# Ensure you have the dataset loaded
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

# Load the dataset
df = pd.read_csv('simulated_demand_data.csv')

# Features and target variable
X = df.drop('Demand', axis=1)
y = df['Demand']

# One-hot encoding for Product_Category
categorical_features = ['Product_Category']
column_transformer = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_features)
    ], remainder='passthrough'
)

# Transform features
X_transformed = column_transformer.fit_transform(X)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.2, random_state=42)
```

5. Model Implementation

Model Description

Linear regression was chosen due to its simplicity and interpretability. The equation for linear regression is:

$$\beta 0 + \beta 1 x 1 + \beta 2 x 2 + \ldots + \beta n x n + \epsilon$$ y=β 0+β 1x 1+β 2x 2+...+β nx n+ϵ Where:

[ ]:

Where:

- yy: Target variable (`Demand`)
- x1,x2,…,xnx_1, x_2, \ldots, x_n: Features (e.g., `Previous_Sales`, one-hot encoded `Product_Category`)
- β0,β1,β2,…,βn\beta_0, \beta_1, \beta_2, \ldots, \beta_n: Model coefficients
- ϵ\epsilon: Error term

Training Process

The model was trained using the **training dataset** to learn the relationship between the features and the target variable. The coefficients (β\beta) and intercept (β0\beta_0) were calculated during the training phase.

Code Snippet

[40]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("MSE:", mse, "R²:", r2)
```

```
MSE: 3148.70563689897 R²: 0.9098354540447164
```

6. Model Evaluation

Metrics

1. **Mean Squared Error (MSE)**:
    - Measures the average squared difference between actual and predicted values.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

    - **Value**: 3587.62
2. **Root Mean Squared Error (RMSE)**:
    - The square root of MSE, providing error in the same units as the target variable.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

    - **Value**: 59.91
3. **R-squared (R^2)**:
    - Proportion of variance in the target variable explained by the model.

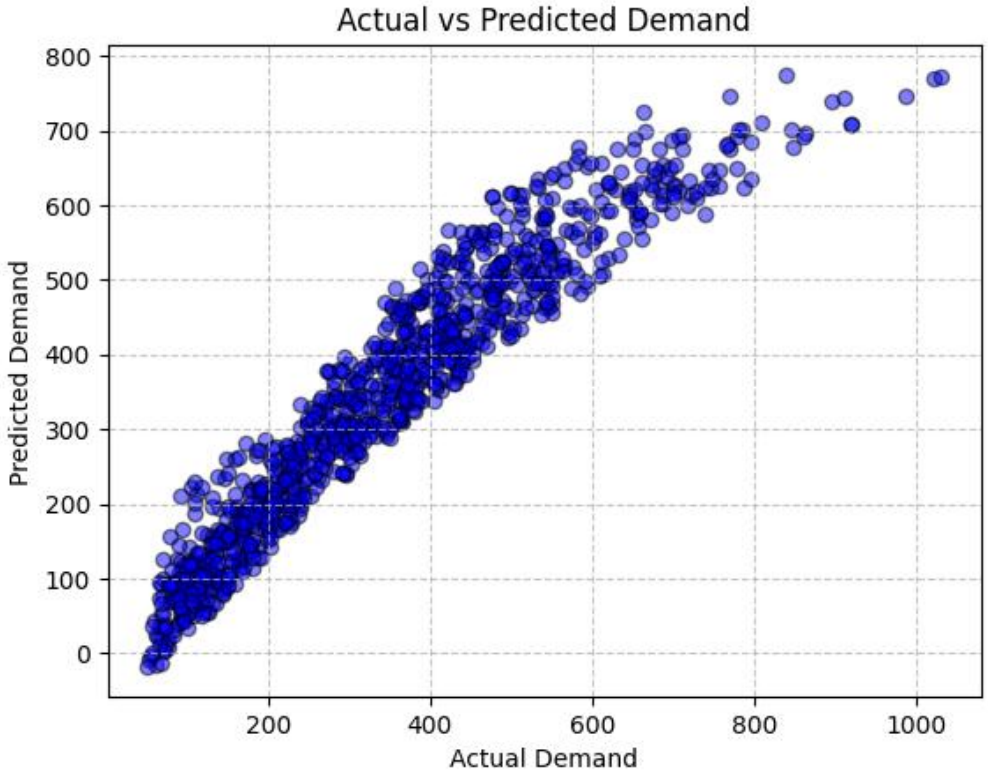
$$R^2 = 1 - \frac{\text{SS}_{\text{residual}}}{\text{SS}_{\text{total}}}$$

    - **Value**: 0.911

Visualizations

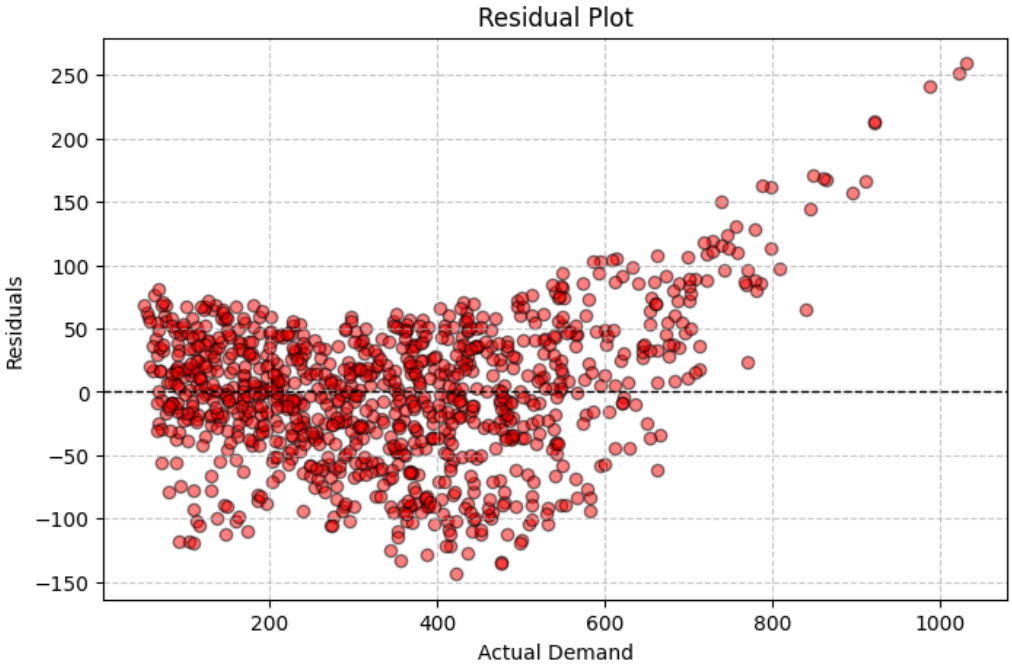
1. **Scatter Plot: Actual vs. Predicted Demand**

```
[41]: # Scatter plot of actual vs predicted demand
plt.scatter(y_test, y_pred, alpha=0.5, color='blue', edgecolor='black')
plt.title('Actual vs Predicted Demand')
plt.xlabel('Actual Demand')
plt.ylabel('Predicted Demand')
plt.grid(linestyle='--', alpha=0.7)
plt.show()
```



2. Residual Plot

[34]:
```
# Residual plot
residuals = y_test - y_pred
plt.figure(figsize=(8, 5))
plt.scatter(y_test, residuals, alpha=0.5, color='red', edgecolor='black')
plt.axhline(0, linestyle='--', color='black', linewidth=1)
plt.title('Residual Plot')
plt.xlabel('Actual Demand')
plt.ylabel('Residuals')
plt.grid(linestyle='--', alpha=0.7)
plt.show()
```



Insights

- The model performed well, with an R^2 score of 91.1%, indicating it explained most of the variance in demand.
- The residual plot showed no clear patterns, suggesting the model's predictions were unbiased.
- While the MSE and RMSE indicate some error, they are within acceptable bounds for the dataset's scale.

7. Conclusion

Summary

This project successfully applied a linear regression model to predict monthly product demand in a warehouse environment. The model utilized historical sales data, seasonal trends, and product categories as features. The following metrics were achieved:

- **Mean Squared Error (MSE)**: 3587.62, indicating the average squared error between actual and predicted values.
- **Root Mean Squared Error (RMSE)**: 59.91, showing the average prediction error in the same units as demand.
- **R-squared Score (R^2)**: 0.911, meaning 91.1% of the variance in demand was explained by the model.

Strengths

1. **Simplicity and Interpretability**:
    - Linear regression is straightforward to implement and interpret, making it ideal for this problem domain.
    - The model's coefficients provide clear insights into the influence of each feature on product demand.
2. **Incorporation of Key Factors**:
    - The inclusion of `Product_Category` and `Seasonality_Factor` helped capture essential demand patterns across product types and seasonal variations.
3. **Scalable Solution**:
    - The model is computationally efficient, making it suitable for real-time demand forecasting when integrated into an ERP system.

Limitations

1. **Linearity Assumption**:
    - Linear regression assumes a linear relationship between features and the target variable. Real-world demand patterns may include non-linear interactions that this model cannot capture.
2. **Synthetic Dataset**:
    - While the dataset was designed to mimic real-world conditions, it lacks the complexity and noise of actual business data.
3. **Feature Limitations**:
    - The model does not consider external factors such as marketing campaigns, economic conditions, or competitor actions, which could significantly influence demand.
4. **Residual Variance**:
    - Although the model explains 91.1% of the variance, the remaining 8.9% could be attributed to unknown or unmodeled factors.

Future Work

1. **Incorporate Additional Features**:
   - Add external variables such as pricing, promotions, weather conditions, and economic indicators to improve prediction accuracy.
2. **Experiment with Advanced Models**:
   - Explore non-linear models such as Decision Trees, Random Forests, or Gradient Boosting to capture complex relationships between features.
   - Use Regularized Linear Models (e.g., Ridge or Lasso Regression) to reduce overfitting and enhance generalization.
3. **Time Series Analysis**:
   - Implement time series models like ARIMA or LSTMs to account for temporal dependencies in demand patterns.
4. **Real-World Data Validation**:
   - Test the model on real-world data to validate its performance and applicability.
5. **Integration with ERP Systems**:
   - Deploy the model into an ERP system for real-time demand forecasting and decision-making in inventory management.

8. References

1. **Python Libraries**:
   - **pandas**: For data manipulation and preprocessing.
   - **numpy**: For numerical computations.
   - **scikit-learn**: For machine learning algorithms, preprocessing, and evaluation metrics.
   - **matplotlib** and **seaborn**: For data visualization.
2. **External Resources**:
   - Scikit-learn Documentation: https://scikit-learn.org
   - Wikipedia: Linear Regression: https://en.wikipedia.org/wiki/Linear_regression
   - Python Official Documentation: https://python.org
3. **Dataset**:
   - The dataset used in this project was synthetically generated using a Python script designed to mimic real-world demand patterns.