GrocerGenius : AI-Based Supermarket Sales Prediction

Report submitted to

Infosys Springboard Team

Mentored By

AMAL SALIAN

PROJECT CONTRIBUTORS(interns)

Aman	**Sayantan Chakrabarti**
Vrushika K Panchal	**Muskan Asthana**
Chetan	**Purnima Pattnaik**
Rimi	**Rameswar Bisoyi**
Shilpa Manaji	**Raunit**
Tharun	**Hima Mankanta**
Sumithra	**Nuka Abhinay**
Yanvi Arora	**Anjan Kumar**

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our mentor at the company, Mr. Amal Salian, for his invaluable guidance, encouragement, and support throughout the successful completion of our internship. His expertise and insights were instrumental in helping us overcome challenges and achieve the desired outcomes.

We would also like to extend our thanks to everyone who contributed, both directly and indirectly, during our internship at Infosys Springboard. Their support and inspiration have been greatly appreciated.

ABSTRACT

Accurate sales prediction is a critical aspect of effective supermarket management, enabling businesses to optimize inventory, plan promotions, and improve customer satisfaction. This study focuses on the development of an Artificial Intelligence (AI) model designed to predict supermarket sales using historical data. The model employs advanced machine learning techniques to analyze complex patterns and relationships within the data, including historical sales trends, seasonal fluctuations, product categories, and promotional impacts.

A comprehensive dataset is preprocessed to address challenges such as missing values, outliers, and categorical variables, ensuring the data's readiness for modeling. Various algorithms, including Random Forest, XG Boosting are evaluated for their predictive performance. The resulting model demonstrates a high degree of accuracy, providing actionable insights into future sales trends.

To enhance usability, an intuitive user interface (UI) is developed, enabling users to interact with the model seamlessly. The UI allows supermarket managers to input specific parameters, visualize predictions, and make data-driven decisions in real-time. By integrating AI-powered forecasting with user-friendly interaction, this solution addresses key challenges in retail operations, offering a scalable and efficient tool for enhancing supermarket performance.

The study highlights the potential of AI in transforming sales prediction, emphasizing its practical implications for improving operational efficiency and customer satisfaction in the retail sector.

CONTENTS

Description **Page-No.**

CHAPTER 1
INTRODUCTION

1.1 Introduction

In the age of digital transformation, data-driven decision-making has become a cornerstone for enhancing business performance and operational efficiency. The retail industry, particularly supermarkets, generates an immense volume of transactional data daily, offering a unique opportunity to leverage artificial intelligence (AI) for predictive analytics. Accurate sales prediction not only enables efficient inventory management but also enhances customer satisfaction and maximizes profitability.

This report introduces *GrocerGenius*, an AI-based solution for supermarket sales prediction, designed to harness the power of machine learning to analyze historical sales data. The primary objective is to develop a robust model capable of forecasting sales trends, providing insights for data-driven decision-making, and offering a user-friendly interface for interaction with predictive outputs.

The internship focuses on a systematic workflow encompassing data collection, exploratory data analysis (EDA), data preprocessing, model development, evaluation, and deployment. Cutting-edge techniques, such as feature engineering and hyperparameter tuning, are applied to enhance model accuracy and generalization. Finally, the model is deployed through a web-based framework to ensure accessibility and practical application.

By presenting a comprehensive approach to supermarket sales prediction, this internship aims to demonstrate the potential of AI in transforming the retail landscape, setting a benchmark for future applications of predictive analytics in similar domains.

1.2 Problem Statement

In the competitive and dynamic supermarket industry, accurately predicting sales is crucial for efficient inventory management, reducing waste, and maximizing profitability. Sales forecasts are influenced by complex and interdependent factors, such as seasonal trends, customer preferences, pricing strategies, and promotional campaigns. However, traditional methods often fail to capture the intricate patterns in retail data, leading to unreliable predictions and operational inefficiencies.

Supermarkets face additional challenges in handling incomplete and noisy datasets, making it difficult to extract actionable insights. The absence of user-friendly predictive tools further limits their ability to implement data-driven decision-making effectively. These shortcomings can result in overstocking or understocking, financial losses, and diminished customer satisfaction.

This research addresses these challenges by developing *GrocerGenius*, an AI-based system for supermarket sales prediction. By leveraging historical sales data and advanced machine learning techniques, the solution aims to provide accurate forecasts and an intuitive interface to enable seamless integration into retail operations, ensuring enhanced efficiency and profitability.

1.3 Objective

The primary objective of this internship project was to conceptualize, design, and implement *GrocerGenius*, an AI-based system aimed at accurately predicting supermarket sales. This work focused on leveraging advanced machine learning techniques to address the multifaceted challenges faced by supermarkets in optimizing inventory, managing resources, and minimizing operational inefficiencies.

1. **Developing an Accurate Predictive Model**: Constructing a robust machine learning model to forecast sales with high precision, incorporating factors such as seasonal trends, customer preferences, and promotional effects.
2. **Building a Comprehensive Data Pipeline**: Designing and implementing a preprocessing framework to handle missing values, perform feature engineering, and manage categorical and numerical data for optimal model performance.
3. **Creating a User-Centric Interface**: Developing an intuitive and interactive user interface that facilitates data upload, prediction visualization, and actionable insights for non-technical stakeholders.

4. **Deploying the System for Practical Use**: Deploying the predictive model using a web-based framework, ensuring ease of accessibility and seamless integration into supermarket operations.
5. **Enabling Data-Driven Decision-Making**: Empowering supermarkets to make informed decisions related to inventory management, staffing, and promotional planning, thereby reducing waste and enhancing profitability

CHAPTER 2

THEORETICAL STUDY

2.1 Theoretical Concepts

2.1.1 Machine Learning and Predictive Analytics

Machine learning (ML) is a subset of artificial intelligence that focuses on building systems capable of learning and improving from experience without explicit programming. In the context of predictive analytics, ML algorithms analyze historical data to identify patterns and trends, enabling accurate forecasts for future events. Supervised learning, a widely used approach in ML, is particularly relevant for sales prediction tasks, as it involves training models on labeled data where the target variable (e.g., sales) is known.

2.1.2 Supermarket Sales Prediction

Supermarket sales prediction relies on leveraging past sales data to forecast future trends. This involves analyzing factors such as seasonal variations, promotional impacts, customer preferences, and economic conditions. The goal is to predict the quantity of goods likely to be sold over a given period, which aids in inventory management, cost optimization, and enhanced customer satisfaction.

2.1.3 Data Processing and Feature Engineering

Data preprocessing is a critical step in preparing raw data for machine learning models. It includes tasks such as handling missing values, removing outliers, encoding categorical variables, and normalizing numerical features. Feature engineering involves creating meaningful input variables (features) that improve model performance by capturing domain-specific insights from the data.

2.1.4 Model Evaluation Metrics

Evaluation metrics play a crucial role in assessing the performance of predictive models. Common metrics for regression problems like sales prediction include:

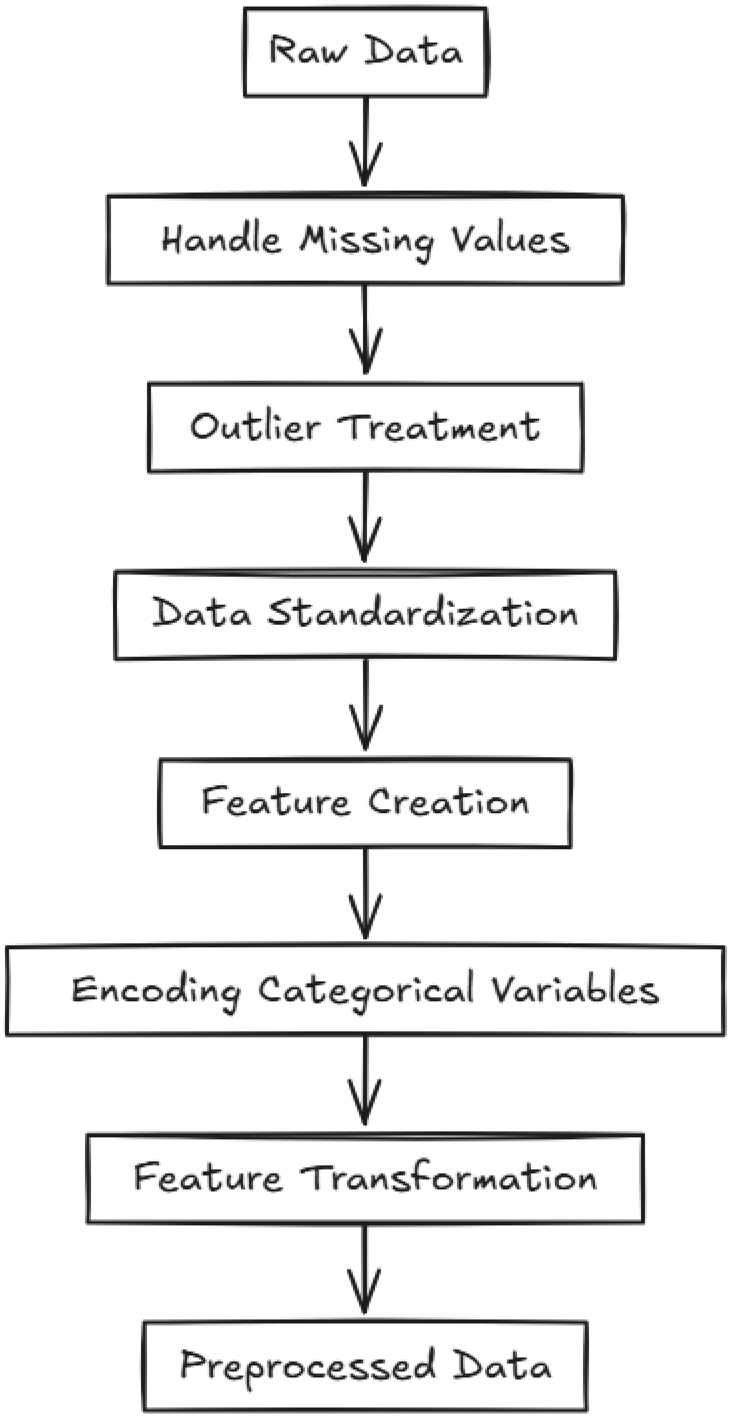
- R-squared (R^2): Represents the proportion of variance in the target variable explained by the model. These metrics help determine the accuracy and reliability of the developed predictive model.

2.1.5 Deployment of Predictive Models

Deployment is the process of integrating a trained model into a production environment where it can provide predictions on new data. This often involves the use of web frameworks, application programming interfaces (APIs), and cloud-based platforms to ensure scalability and accessibility.

CHAPTER 3

ARCHITECTURE/DESIGN



System Architecture Overview

1. Data Layer

- Storage and Management:
  - Historical sales data from supermarkets, including product categories, sales transactions, and promotional information, is stored in CSV files.
  - Data Management:
    - Data is processed using Python libraries such as pandas and NumPy to load, clean, and explore the dataset.

2. Processing Layer

- Data Cleaning and Preprocessing:
  - Python libraries like pandas and scikit-learn are used to clean the data, handle missing values, and perform feature engineering. For example, categorical features (e.g., product types) are encoded, and numerical features (e.g., sales figures) are normalized.
- Feature Engineering:
  - Relevant features such as seasonal trends, product popularity, and promotional effectiveness are created to enhance prediction accuracy. Additional features like time-based aggregates (e.g., monthly or weekly sales) are also generated.
- Modeling:
  - Machine learning models, including Random Forest, Gradient Boosting (e.g., XGBoost) are trained on the processed data to predict supermarket sales.

    - Cross-validation and hyperparameter tuning (using tools like GridSearchCV or RandomizedSearchCV) are employed to improve model performance.
- Model Evaluation:
    - The models are evaluated using regression metrics such as R-squared (R^2). The best-performing model is selected for deployment and saved using libraries like joblib or pickle.

3. Presentation Layer

- Web Interface:
    - A web application is developed using frameworks like steamlit to enable user interaction. The interface supports:
        - Uploading new sales data.
        - Displaying predictions for future sales.
        - Visualizing trends and model insights.
- User Interaction:
    - Users can input parameters such as product categories, promotional details, or time periods and receive predictions in real time.

4.Key Design Decisions

- Web Framework Selection:
    - Stemlit was chosen for their robust backend capabilities and ease of integration with machine learning models.
- Data Format:
    - CSV files are used initially for simplicity and portability of data handling. For larger datasets or multi-user environments, transitioning to a relational database is planned.

5.Trade-offs

- Simplicity vs. Scalability:
    - The choice of CSV files simplifies data management during the development phase but limits scalability for larger datasets.
- Model Complexity:
    - Advanced models like Neural Networks were considered but avoided for initial deployment to maintain computational efficiency and interpretability.

Development

Technologies Used

- Streamlit: For creating the front-end interface and deploying the web application.
- Scikit-learn: For implementing machine learning algorithms and predictive modeling.
- Pandas & NumPy: For efficient data manipulation, cleaning, and analysis.
- VS Code & Jupyter Notebook: Used for writing, testing, and analyzing Python code during development.

Coding Standards

- Python Scripting with VS Code: Ensuring clean and organized code through VS Code for better debugging and collaboration.
- Modular Code Structure: Emphasizing a modular approach to enhance scalability, maintainability, and readability of the codebase.

Challenges

- Integration: Ensuring seamless interaction between Streamlit and Plotly for a smooth user experience.
- Model Optimization: Fine-tuning the performance of machine learning models to achieve accurate sales predictions using R^2 as the primary evaluation metric.
- Data Consistency: Addressing inconsistencies in data formatting and handling missing or noisy data effectively.

Solutions

- Standardized Preprocessing Pipeline: Developed a robust pipeline to ensure consistent data inputs for the machine learning models.
- Iterative Testing and Feedback Loops: Continuously improved the machine learning models and user interface through iterative testing and user feedback.
- Data Validation and Cleaning: Implemented systematic data validation and cleaning procedures to resolve inconsistencies and ensure high-quality data for analysis.

CHAPTER 4

DEPLOYMENT

Process:

1. Host the Streamlit App on Streamlit Cloud
   Deploy the Streamlit app on Streamlit Cloud to make the Grocery Sales Prediction interface publicly accessible for users to interact with and generate sales predictions.
2. Integrate Machine Learning Pipeline into Streamlit
   Incorporate the trained machine learning model (Random Forest Regressor) into the Streamlit app to predict grocery item sales. The app will process user inputs such as item details, outlet information, and pricing to provide accurate sales predictions.
3. Forecasting using Historical Data
   Utilize the machine learning pipeline to forecast sales for various grocery items based on historical sales data trends, allowing businesses to optimize inventory and pricing strategies.
4. Ensure Model Efficiency and Deployment
   Train and save the machine learning model using libraries like Joblib for efficient storage. Load the model in the Streamlit app to enable real-time sales predictions with minimal latency.

Automation:

- Deployment Scripts:
  Create deployment scripts (e.g., Streamlit Cloud configuration files) to automate the app update process, ensuring seamless transitions between local development and the cloud-hosted environment.
  Set up an automated deployment process to reflect model updates and pipeline improvements directly in the live app.

Instructions for Deployment: Grocery Sales Prediction Project

Clone the Repository

Clone the project repository to your local machine or remote environment using the following command:

```
git clone
https://github.com/yourusername/grocery_sales_predicti
on.git
cd grocery_sales_prediction
```

1.

Set up Environment

Create and activate a virtual environment for the project:

```
python3 -m venv env
source env/bin/activate  # For Windows:
env\Scripts\activate
```

2.

Install Dependencies
Install the required Python libraries using the `requirements.txt` file:

```
pip install -r requirements.txt
```

3.

Prepare Model Files
Ensure the trained Random Forest model (e.g., Joblib file) is placed in the designated directory `(model_factory/models/)` for the Streamlit app to load during execution.

Run Streamlit App Locally
Test the app on your local machine to ensure everything works correctly

```
streamlit run codebase/app.py
```

CHAPTER 5

CONCLUSION

The **Grocery Sales Prediction Project** successfully provides:

- **Interactive Dashboards:**
  The Streamlit-based User Interface (UI) allows users to input product and outlet details to forecast sales. It visualizes predicted sales trends, highlights key features driving sales, and provides comparative insights across outlets. The intuitive design ensures a seamless experience for both retail managers and analysts.

- **AI-powered Insights:**
  By leveraging the Random Forest model, the app generates precise sales predictions. It analyzes historical data to provide actionable insights for inventory management, pricing strategies, and demand forecasting, enabling businesses to make informed decisions.

- **Custom Reporting:**
  Tailored predictions cater to various needs, such as optimizing stock levels, planning promotions, and setting outlet-specific targets. Interactive charts and reports help explore trends and patterns effectively.

Lessons Learned

- **Early Integration Testing Prevents Deployment Issues:**
  Integrating the machine learning model, preprocessing pipeline, and Streamlit app early in development avoided potential bottlenecks and ensured a smooth user experience.

- **User Feedback is Invaluable:**
  Insights from end-users, such as store managers and analysts, helped refine the app's usability, visualizations, and prediction accuracy, making it more impactful.

- **Model Training and Evaluation Are Key:**
  Experimenting with different features and tuning hyperparameters improved the model's performance, ensuring reliable and accurate sales predictions.

Future Enhancements

- **Integrating Advanced Predictive Analytics:**
  Incorporate more advanced models (e.g., Gradient Boosting, Time Series Forecasting) to capture seasonal trends and complex sales patterns for improved accuracy.

- **Adding Real-Time Data Updates:**
  Enable real-time data integration for live sales forecasting, allowing

the app to adapt dynamically to market conditions.

- **Enhancing User Experience:**
  Add features like personalized recommendations for outlet-specific strategies, visualizing seasonal trends, or generating detailed comparisons between different outlets and products.

CHAPTER 6

REFERENCES

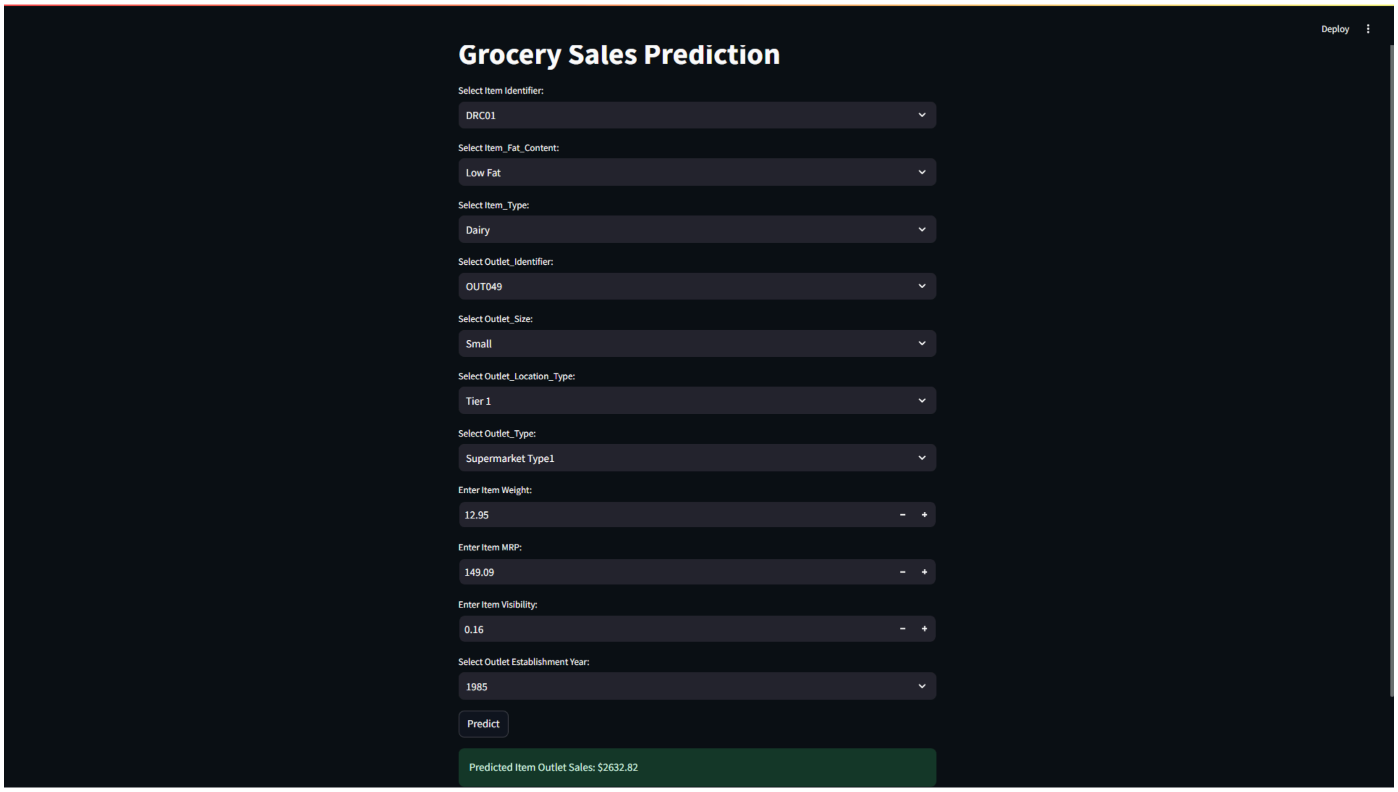
1. **Pandas Documentation:**
   A powerful Python library for data manipulation and analysis.
   https://pandas.pydata.org/docs/
2. **NumPy Documentation:**
   The fundamental library for numerical computations in Python.
   https://numpy.org/doc/
3. **Scikit-learn Documentation:**
   Comprehensive machine learning tools for Python.
   https://scikit-learn.org/stable/documentation.html
4. **Streamlit Documentation:**
   Framework for creating data applications.
   https://docs.streamlit.io/
5. **Matplotlib Documentation:**
   Python library for creating static, animated, and interactive visualizations.
   https://matplotlib.org/stable/contents.html
6. **Seaborn Documentation:**
   Python visualization library for statistical graphics.
   https://seaborn.pydata.org/
7. **Random Forest Algorithm (Breiman, L., 2001):**
   Random forests. *Machine Learning, 45(1),* 5-32.
   https://link.springer.com/article/10.1023/A:1010933404324

CODE AND SNAPSHOTS

```
# codebase/training_script.py

import os
import pandas as pd
from utils import preprocess_data, train_model

# Define directories
BASE_DIR =
os.path.dirname(os.path.dirname(os.path.abspath
(__file__)))
DATA_DIR = os.path.join(BASE_DIR,
'data_alchemy', 'raw')

# Load your training data
df = pd.read_csv(os.path.join(DATA_DIR,
'train.csv'))

# Preprocess and train
preprocessed_data = preprocess_data(df,
is_training=True)
preprocessed_data['Item_Outlet_Sales'] =
df['Item_Outlet_Sales'].values
model = train_model(preprocessed_data)
```

```
# codebase/app.py

import os
import streamlit as st
import pandas as pd
from utils import run_inference

# Set the base directory
BASE_DIR =
os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

st.title("Grocery Sales Prediction")

# Load unique values for Item_Identifier (optional)
item_identifiers = ['FDA15', 'DRC01', 'FDN15', 'FDX07', 'NCD19']  #
Replace with actual identifiers if needed

# Dropdown options for categorical columns (excluding
Item_Identifier)
dropdown_options = {
    "Item_Fat_Content": ["Low Fat", "Regular"],
    "Item_Type": [
        "Dairy", "Soft Drinks", "Meat", "Fruits and Vegetables",
        "Household", "Baking Goods", "Snack Foods", "Frozen Foods",
        "Breakfast", "Health and Hygiene", "Hard Drinks", "Canned",
        "Breads", "Starchy Foods", "Others", "Seafood"
    ],
    "Outlet_Identifier": ["OUT049", "OUT018", "OUT010", "OUT013",
"OUT027", "OUT045", "OUT017", "OUT046", "OUT035", "OUT019"],
    "Outlet_Size": ["Small", "Medium", "High"],
    "Outlet_Location_Type": ["Tier 1", "Tier 2", "Tier 3"],
    "Outlet_Type": [
        "Supermarket Type1", "Supermarket Type2", "Supermarket
Type3", "Grocery Store"
    ],
}

# Numerical ranges for inputs
value_ranges = {
    "Item_Visibility": (0.0, 0.328391),  # Min, Max from data
    "Item_Weight": (4.555, 21.35),
    "Item_MRP": (31.29, 266.8884),
}

# Collect user inputs
user_inputs = {}

# Dropdown for Item Identifier (if needed)
user_inputs["Item_Identifier"] = st.selectbox(
    "Select Item Identifier:", options=item_identifiers
)

# Dropdowns for categorical inputs
for key, options in dropdown_options.items():
    user_inputs[key] = st.selectbox(f"Select {key}:", options)

# Number input for Item Weight
user_inputs["Item_Weight"] = st.number_input(
    "Enter Item Weight:", min_value=value_ranges["Item_Weight"][0],
max_value=value_ranges["Item_Weight"][1],
    value=(value_ranges["Item_Weight"][0] +
value_ranges["Item_Weight"][1]) / 2, step=0.01
)

# Number input for Item MRP
user_inputs["Item_MRP"] = st.number_input(
    "Enter Item MRP:", min_value=value_ranges["Item_MRP"][0],
max_value=value_ranges["Item_MRP"][1],
    value=(value_ranges["Item_MRP"][0] + value_ranges["Item_MRP"][1])
/ 2, step=0.01
)

# Number input for Item Visibility
user_inputs["Item_Visibility"] = st.number_input(
    "Enter Item Visibility:",
min_value=value_ranges["Item_Visibility"][0],
max_value=value_ranges["Item_Visibility"][1],
    value=(value_ranges["Item_Visibility"][0] +
value_ranges["Item_Visibility"][1]) / 2, step=0.0001
)

# Selectbox for Outlet Establishment Year
years = list(range(1985, 2010))  # Years from 1985 to 2009
user_inputs["Outlet_Establishment_Year"] = st.selectbox(
    "Select Outlet Establishment Year:", options=years
)

# When the user clicks the "Predict" button
if st.button("Predict"):
    # Convert user_inputs dictionary to DataFrame
    user_input_df = pd.DataFrame([user_inputs])

    # Run inference
    try:
        predictions = run_inference(user_input_df)
        predicted_sales = predictions[0]

        st.success(f"Predicted Item Outlet Sales:
${predicted_sales:.2f}")
    except Exception as e:
        st.error(f"Error during prediction: {e}")
```

UI SNAPSHOT1



Grocery Sales Prediction

Select Item Identifier: DRC01

Select Item_Fat_Content: Low Fat

Select Item_Type: Dairy

Select Outlet_Identifier: OUT049

Select Outlet_Size: Small

Select Outlet_Location_Type: Tier 1

Select Outlet_Type: Supermarket Type1

Enter Item Weight: 12.95

Enter Item MRP: 149.09

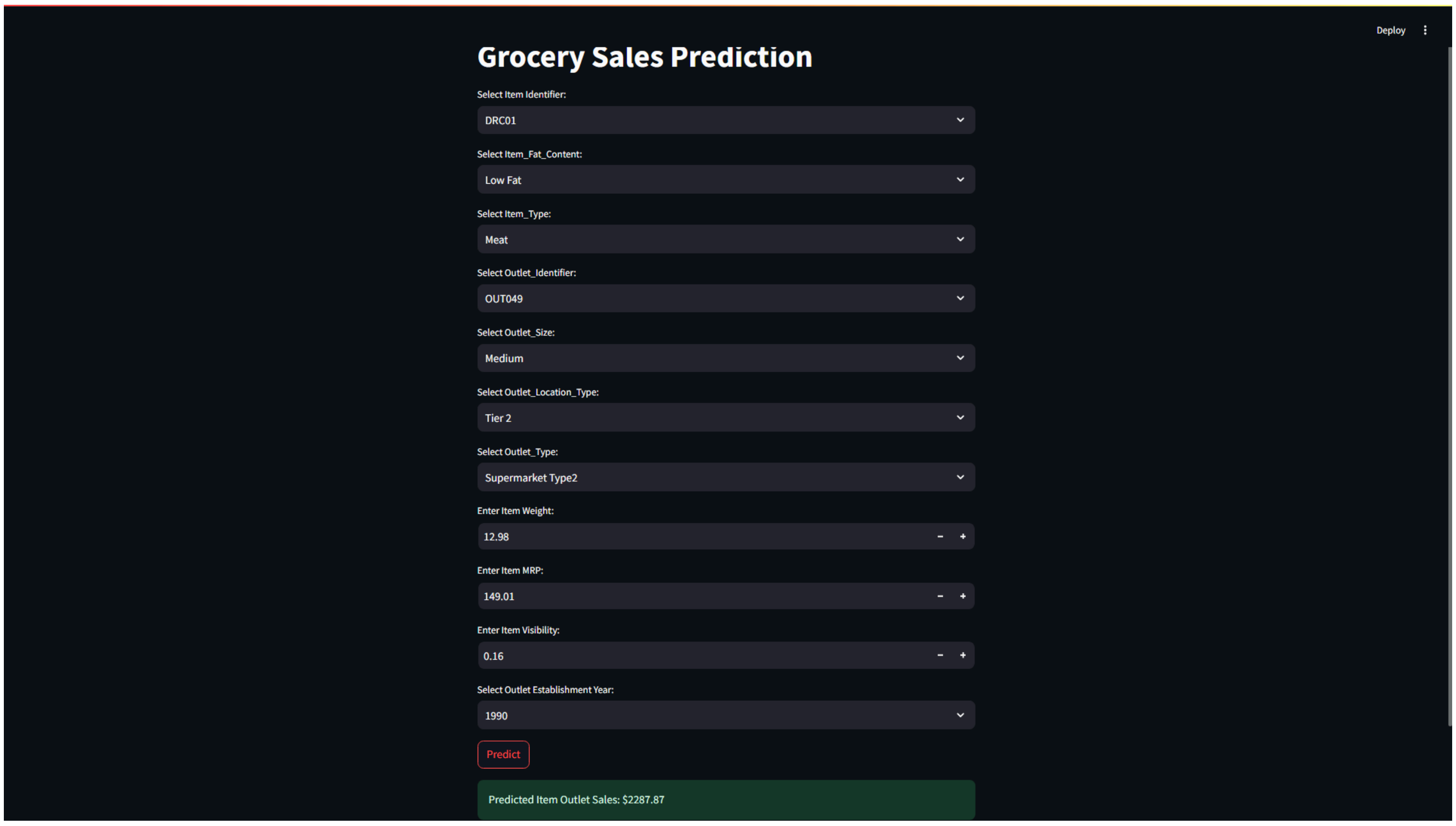
Enter Item Visibility: 0.16

Select Outlet Establishment Year: 1985

Predict

Predicted Item Outlet Sales: $2632.82

UI SNAPSHOT2



Grocery Sales Prediction

Select Item Identifier:
DRC01

Select Item_Fat_Content:
Low Fat

Select Item_Type:
Meat

Select Outlet_Identifier:
OUT049

Select Outlet_Size:
Medium

Select Outlet_Location_Type:
Tier 2

Select Outlet_Type:
Supermarket Type2

Enter Item Weight:
12.98

Enter Item MRP:
149.01

Enter Item Visibility:
0.16

Select Outlet Establishment Year:
1990

Predict

Predicted Item Outlet Sales: $2287.87