Student Name: P.P.SAKTHIVEL
Register Number: C3S20570
Institution: PKN ARTS AND SCIENCE COLLEGE
Department: INFORMATION TECHNOLOGY
Date of Submission: 19/01/2026
Github Repository Link:
https://github.com/Sakthi1509/Restaurant-Sales-Data-Analysis-And-Insights

Project Title: Restaurant Sales Data Analysis and Insights

1. Problem Statement

Restaurants generate large volumes of transactional data daily, but much of this data remains underutilized for decision-making. Accurately predicting the total order value based on order details such as item category, price, quantity, and payment method can help restaurant owners optimize pricing strategies, manage inventory, and forecast revenue. This project addresses the problem of predicting restaurant order totals using historical sales data. It is a **regression problem**, as the target variable (Order Total) is continuous and numeric.

2. Abstract

This project focuses on analyzing restaurant sales data and building a machine learning model to predict the total order value. The dataset consists of historical restaurant transactions containing both categorical and numerical features. The data was cleaned, preprocessed, and explored using visual analytics to identify patterns and relationships. A Linear Regression model was trained on the processed data to predict order totals. The model performance was evaluated using Mean Squared Error (MSE) and R^2 score. Finally, the trained model was deployed using Gradio to provide an interactive web-based prediction interface.

3. System Requirements

- **Hardware Requirements:** - Minimum 4 GB RAM - Any modern processor (Intel i3 or equivalent)

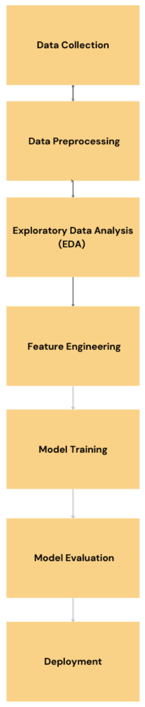
- **Software Requirements:** - Python 3.8 or above - Google Colab / Jupyter Notebook / VS Code - Required Libraries: - pandas - numpy - matplotlib - seaborn - scikit-learn - gradio

4. Objectives

- To analyze restaurant sales data and understand key patterns influencing order value.
- To preprocess and clean raw sales data for machine learning.
- To build a regression model that predicts the total order amount.
- To evaluate the model using standard regression metrics.
- To deploy the trained model as an interactive application for real-time predictions.

5. Flowchart of Project Workflow

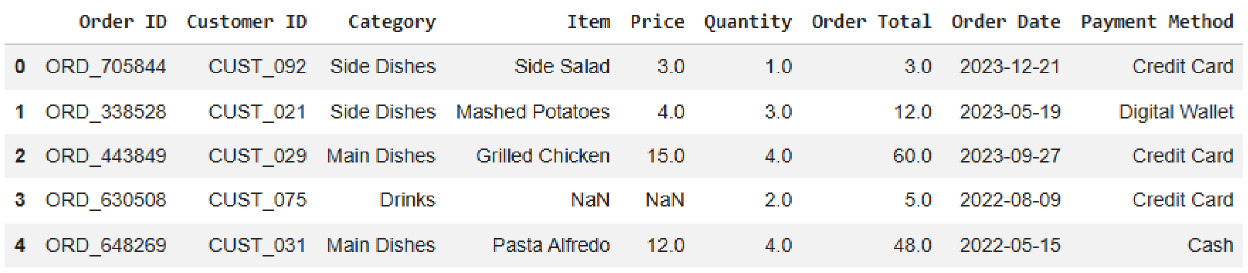
Workflow Steps: 1. Data Collection 2. Data Preprocessing 3. Exploratory Data Analysis (EDA) 4. Feature Engineering 5. Model Training 6. Model Evaluation 7. Deployment



6. Dataset Description

- **Source:** Public open-source dataset (Kaggle-style restaurant sales dataset)
- **Type:** Public dataset
- **Format:** CSV
- **Size:** 17,534 rows × 9 columns

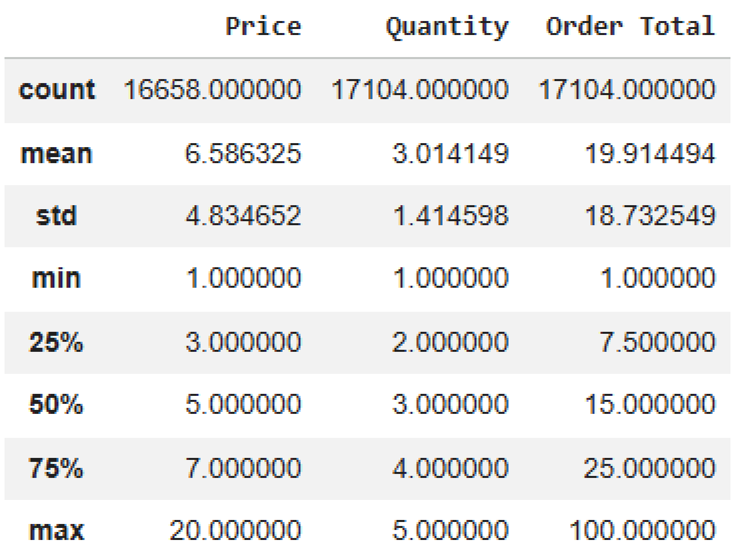
Key Columns: - Category - Item - Price - Quantity - Payment Method - Order Total (Target Variable)



	Order ID	Customer ID	Category	Item	Price	Quantity	Order Total	Order Date	Payment Method
0	ORD_705844	CUST_092	Side Dishes	Side Salad	3.0	1.0	3.0	2023-12-21	Credit Card
1	ORD_338528	CUST_021	Side Dishes	Mashed Potatoes	4.0	3.0	12.0	2023-05-19	Digital Wallet
2	ORD_443849	CUST_029	Main Dishes	Grilled Chicken	15.0	4.0	60.0	2023-09-27	Credit Card
3	ORD_630508	CUST_075	Drinks	NaN	NaN	2.0	5.0	2022-08-09	Credit Card
4	ORD_648269	CUST_031	Main Dishes	Pasta Alfredo	12.0	4.0	48.0	2022-05-15	Cash

7. Data Preprocessing

- Missing values in numerical columns were filled using median values.
- Missing values in categorical columns were filled using mode.
- Duplicate records were removed.
- Irrelevant columns such as Order ID, Customer ID, and Order Date were dropped.
- Categorical variables were converted into numerical form using One-Hot Encoding.
- Feature scaling was applied using StandardScaler.

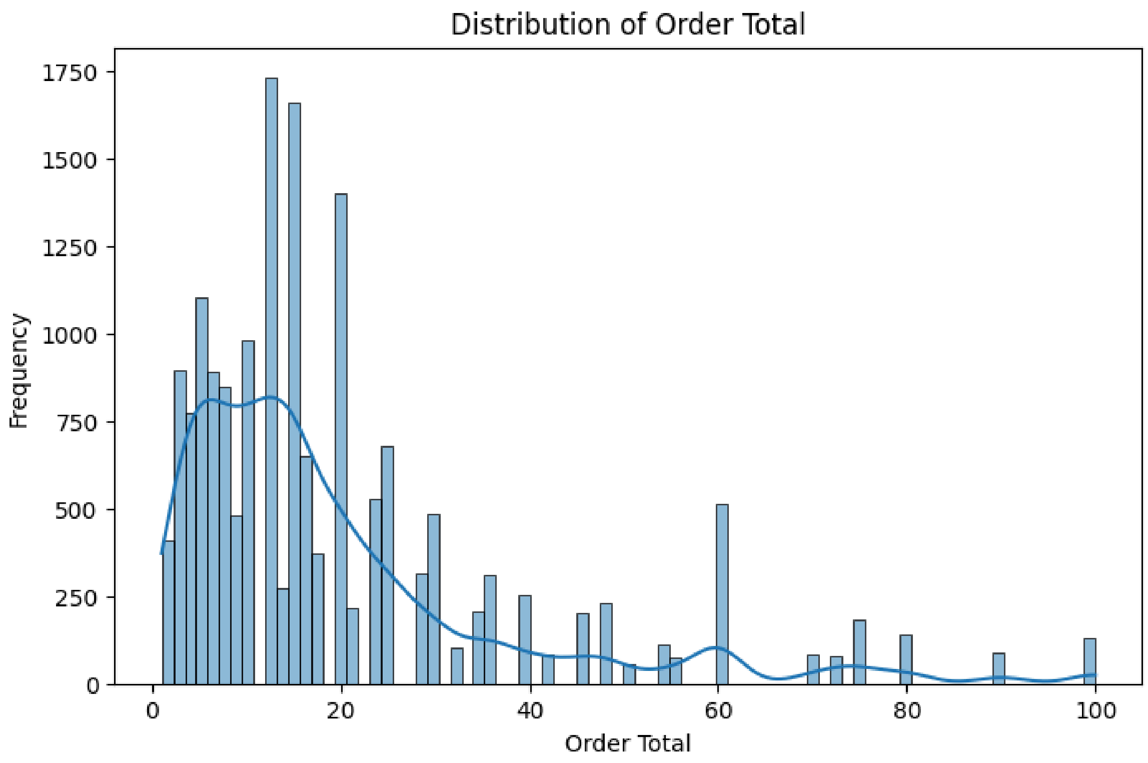


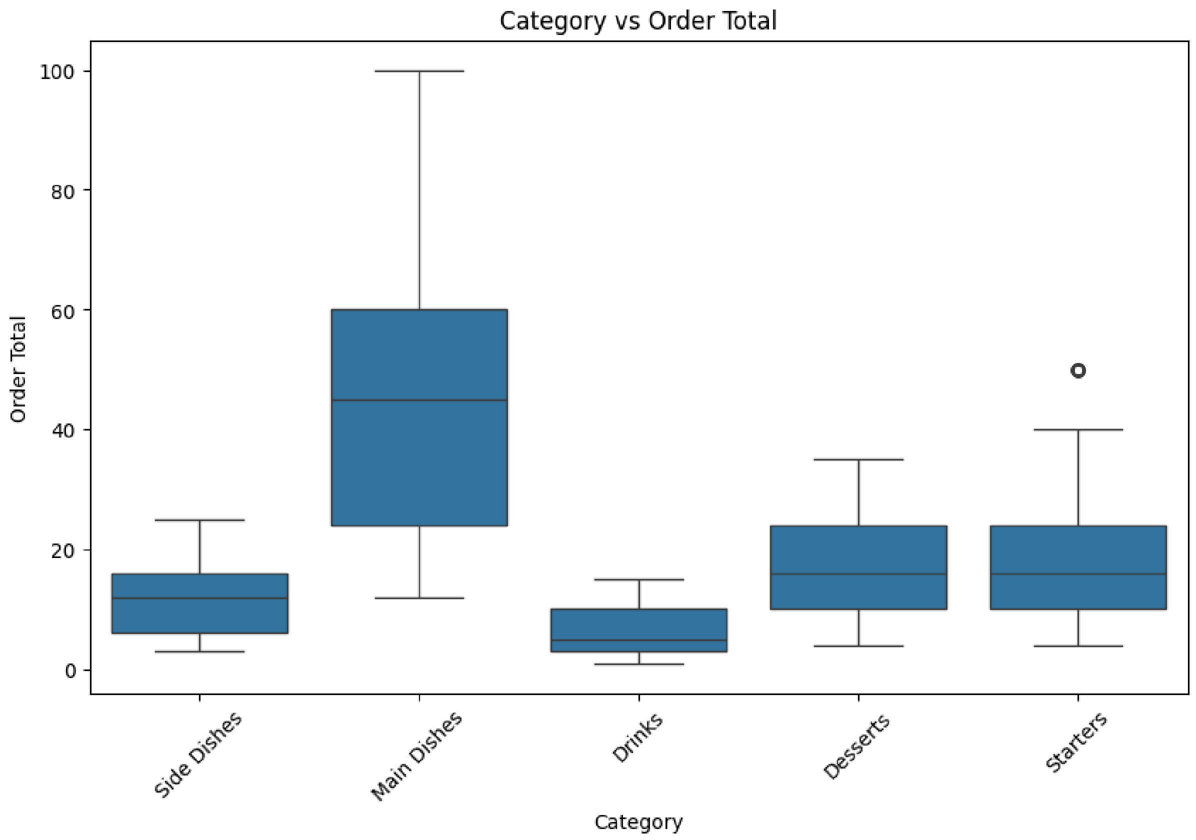
	Price	Quantity	Order Total
count	16658.000000	17104.000000	17104.000000
mean	6.586325	3.014149	19.914494
std	4.834652	1.414598	18.732549
min	1.000000	1.000000	1.000000
25%	3.000000	2.000000	7.500000
50%	5.000000	3.000000	15.000000
75%	7.000000	4.000000	25.000000
max	20.000000	5.000000	100.000000

8. Exploratory Data Analysis (EDA)

- Histogram used to analyze the distribution of Order Total.
- Boxplots used to compare sales across categories and payment methods.
- Scatter plots used to study relationships between Price, Quantity, and Order Total.

Key Insights: - Higher quantities and prices lead to higher order totals. - Certain food categories generate higher average sales. - Payment methods show variation in spending b

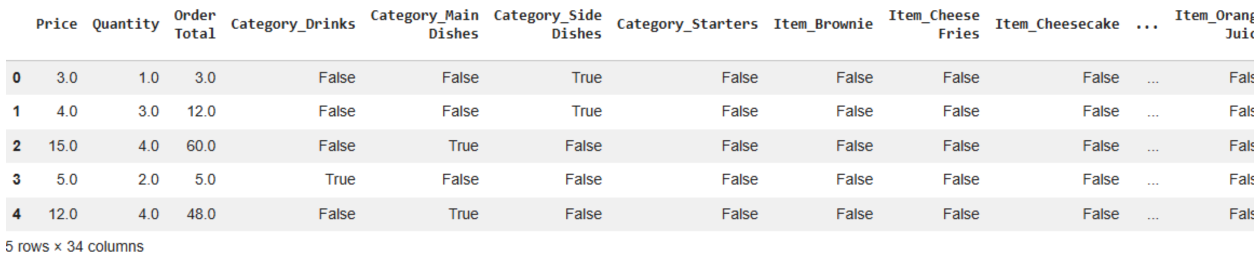

Distribution of Order Total
Frequency
0
250
500
750
1000
1250
1500
1750
0
20
40
60
80
100
Order Total


Category vs Order Total
Order Total
0
20
40
60
80
100
Side Dishes
Main Dishes
Drinks
Desserts
Starters
Category

9. Feature Engineering

- Removal of non-informative identifier columns.
- One-Hot Encoding applied to categorical features.
- Feature scaling to normalize numeric values.

These transformations help improve model performance and ensure compatibility with Linear Regression.



	Price	Quantity	Order Total	Category_Drinks	Category_Main Dishes	Category_Side Dishes	Category_Starters	Item_Brownie	Item_Cheese Fries	Item_Cheesecake	...	Item_Orang Juic
0	3.0	1.0	3.0	False	False	True	False	False	False	False	...	Fals
1	4.0	3.0	12.0	False	False	True	False	False	False	False	...	Fals
2	15.0	4.0	60.0	False	True	False	False	False	False	False	...	Fals
3	5.0	2.0	5.0	True	False	False	False	False	False	False	...	Fals
4	12.0	4.0	48.0	False	True	False	False	False	False	False	...	Fals

5 rows × 34 columns

10. Model Building

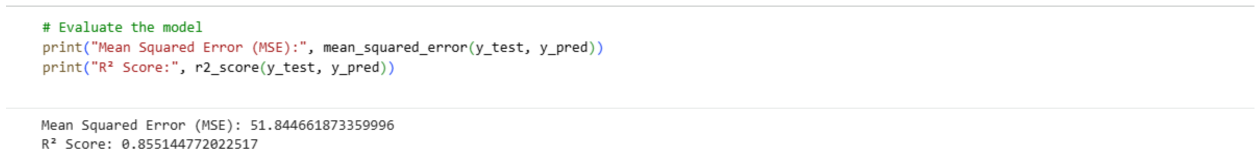
- **Baseline Model:** Linear Regression
- **Reason for Selection:**
  - Suitable for continuous target variables
  - Simple and interpretable
  - Efficient for large tabular datasets

The model was trained using an 80:20 train-test split.

11. Model Evaluation

- **Metrics Used:**
  - Mean Squared Error (MSE)
  - R² Score

The evaluation metrics indicate how well the model predicts unseen data.



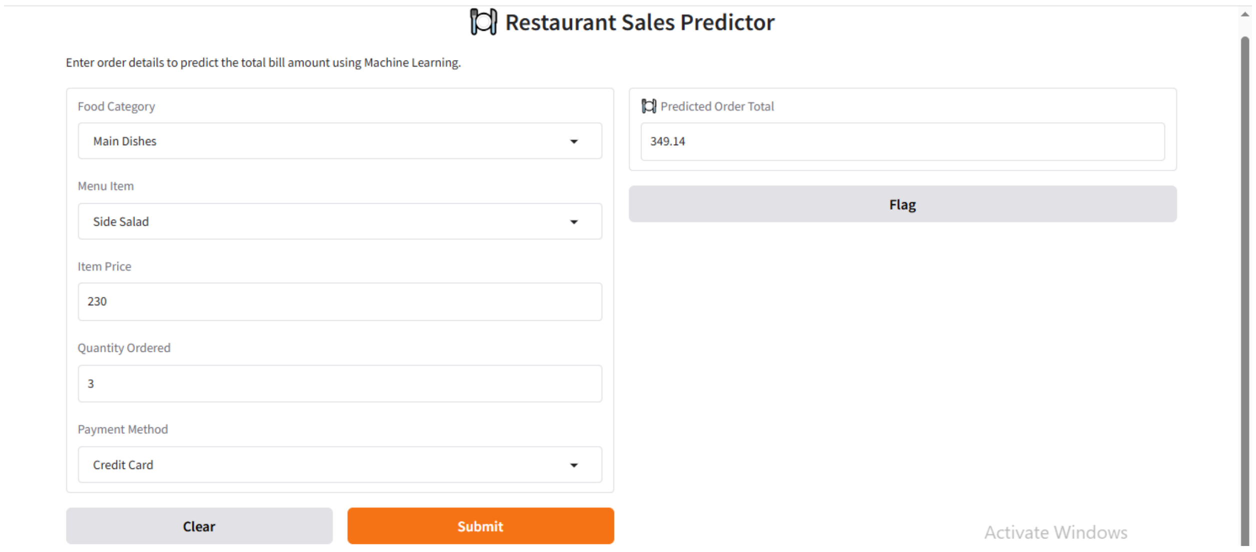
```
# Evaluate the model
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error (MSE): 51.844661873359996
R² Score: 0.855144772022517
```

12. Deployment

- **Deployment Tool:** Gradio interface
- **Public Link:** https://bbd373731c5691d3d9.gradio.live/
- **Method:** Local/Colab-based web interface

- **Features:**
    - User input form
    - Real-time order total prediction



Restaurant Sales Predictor

Enter order details to predict the total bill amount using Machine Learning.

Food Category: Main Dishes

Menu Item: Side Salad

Item Price: 230

Quantity Ordered: 3

Payment Method: Credit Card

Predicted Order Total: 349.14

13. Source Code

```
# project.py

# Restaurant Sales Data Analysis and Insights

# Predict Order Total using Linear Regression & Deploy with Gradio


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean_squared_error, r2_score


import gradio as gr
```

```
# STEP 2: Load Dataset
df = pd.read_csv('restaurant_sales_data.csv')

# STEP 3: EDA & Basic Info
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
df.info()
print(df.describe())

# STEP 4: Data Cleaning
# Fill missing values
numeric_cols = df.select_dtypes(include=['int64','float64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

for col in numeric_cols:
    df[col].fillna(df[col].median(), inplace=True)
for col in categorical_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Remove duplicates
df.drop_duplicates(inplace=True)

# Drop irrelevant columns
df.drop(['Order ID','Customer ID','Order Date'], axis=1, inplace=True)

# STEP 5: Visualizations (EDA)
plt.figure(figsize=(8,5))
```

```
sns.histplot(df['Order Total'], kde=True)
plt.title('Distribution of Order Total')
plt.xlabel('Order Total')
plt.show()

plt.figure(figsize=(10,6))
sns.boxplot(x='Category', y='Order Total', data=df)
plt.title('Category vs Order Total')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x='Payment Method', y='Order Total', data=df)
plt.title('Payment Method vs Order Total')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(6,5))
sns.scatterplot(x='Price', y='Order Total', data=df)
plt.title('Price vs Order Total')
plt.xlabel('Item Price')
plt.ylabel('Order Total')
plt.show()

plt.figure(figsize=(6,5))
sns.scatterplot(x='Quantity', y='Order Total', data=df)
plt.title('Quantity vs Order Total')
plt.xlabel('Quantity')
plt.ylabel('Order Total')
plt.show()

# STEP 6: Target & Features + Encoding
target = 'Order Total'
features = df.columns.drop(target)

categorical_cols = df.select_dtypes(include=['object']).columns
df_encoded = pd.get_dummies(df, drop_first=True)

X = df_encoded.drop(target, axis=1)
y = df_encoded[target]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# STEP 7: Train-Test Split & Model
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

model = LinearRegression()
model.fit(X_train, y_train)

# Predictions & Evaluation
y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
# STEP 8: Predict New Input
new_order = {
    'Category': 'Main Course',
    'Item': 'Chicken Biryani',
    'Price': 250.0,
    'Quantity': 2,
    'Payment Method': 'Credit Card'
}

new_df = pd.DataFrame([new_order])
df_temp = pd.concat([df.drop(target, axis=1), new_df], ignore_index=True)
df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
df_temp_encoded = df_temp_encoded.reindex(columns=X.columns, fill_value=0)
new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
predicted_sales = model.predict(new_input_scaled)
print("Predicted Order Total:", round(predicted_sales[0],2))

# STEP 9: Deployment with Gradio
def predict_order_total(Category, Item, Price, Quantity, Payment_Method):
    input_data = {
        'Category': Category,
        'Item': Item,
        'Price': float(Price),
        'Quantity': float(Quantity),
        'Payment Method': Payment_Method
    }
    input_df = pd.DataFrame([input_data])
    df_temp = pd.concat([df.drop(target, axis=1), input_df], ignore_index=True)
    df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
    df_temp_encoded = df_temp_encoded.reindex(columns=X.columns, fill_value=0)
    scaled_input = scaler.transform(df_temp_encoded.tail(1))
    prediction = model.predict(scaled_input)
    return round(prediction[0],2)

inputs = [
    gr.Dropdown(df['Category'].unique().tolist(), label="Food Category"),
    gr.Dropdown(df['Item'].unique().tolist(), label="Menu Item"),
    gr.Number(label="Item Price"),
    gr.Number(label="Quantity Ordered"),
    gr.Dropdown(df['Payment Method'].unique().tolist(), label="Payment Method")
]

output = gr.Number(label="Predicted Order Total")

gr.Interface(
    fn=predict_order_total,
    inputs=inputs,
    outputs=output,
    title="Restaurant Sales Predictor",
    description="Enter order details to predict the total bill amount."
).launch()
```

14. Future Scope

- Implement advanced models such as Random Forest or XGBoost.
- Include time-based features for sales forecasting.
- Deploy the application on a cloud platform for public access.
- Integrate real-time data streaming from POS systems