Customer Prediction for Pizza Orders - Analysis Report

This report provides an in-depth analysis of customer behavior and purchasing trends for pizza orders. The analysis aims to identify patterns, key insights, and recommendations to enhance customer retention and maximize revenue.

1. Dataset Overview

The dataset consists of 5000 records of pizza orders, including customer demographics, order frequency, pizza preferences, spending habits, and delivery details. Key attributes include:

- Customer Age, Gender, and Income

- Order Frequency and Spending per Order

- Preferred Pizza Type and Size

- Ordering Time and Day of the Week

- Loyalty Membership and Discount Usage

- Delivery Time and Customer Ratings

2. Key Findings

2.1 Customer Age Distribution

Most customers fall between 25 to 55 years old, with a peak around 30-45 years old. This suggests that the primary market for pizza sales is working professionals and young families.

2.2 Income Distribution

The majority of customers have an income range between $30,000 to $80,000, indicating a middle-class audience. Understanding income levels helps in setting pricing strategies and promotional discounts.

2.3 Popular Pizza Types & Sizes

The most ordered pizzas are Pepperoni, Cheese, and Meat Lovers, while Hawaiian is the least preferred. Medium and Large pizzas dominate the orders, indicating a preference for family or group dining.

2.4 Order Frequency by Age Group

Younger customers (18-35) order more frequently than older customers. Businesses can focus marketing efforts on this age group to drive more sales.

2.5 Spending Behavior with Discounts

Customers using discounts tend to spend less per order than those who don't. This suggests that while discounts attract buyers, they may reduce overall revenue per transaction.

2.6 Loyalty Program Insights

Loyalty members tend to spend more per order, making them valuable customers. Encouraging more customers to join the loyalty program can increase long-term revenue.

2.7 Delivery Time and Customer Satisfaction

Longer delivery times correlate with lower customer ratings. Ensuring deliveries are completed in under 30 minutes can improve customer satisfaction and retention.

3. Business Recommendations

1. **Optimize Delivery Efficiency**: Focus on reducing delivery times to improve customer satisfaction and ratings.

2. **Leverage Loyalty Programs**: Promote loyalty memberships as members tend to spend more per order.

3. **Target Younger Audiences**: Design marketing campaigns targeting customers aged 18-35 who order frequently.

4. **Reevaluate Discount Strategy**: Find a balance between discounts and maintaining profitability.

5. **Introduce Combo Deals**: Encourage customers to buy larger-sized pizzas or multiple items to increase revenue per order.

Pizza order datasets

Customer_ID	Age	Gender	Income	Order_Frequency	Preferred_Pizza_Type	Size	Spending_Per_Order	Time_of_Order	Day_of_Week	Loyalty_Member	Discount_Used	Delivery_Time_Minutes	Customer_Rating	Repeat_Customer
10001	56	Female	89374	8	BBQ Chicken	Large	31.73	Evening	Weekday	No	Yes	52	5	Yes
10002	46	Non-binary	87509	3	Hawaiian	Large	13.09	Afternoon	Weekend	No	Yes	55	2	No
10003	32	Female	95783	5	Pepperoni	Medium	27.70	Afternoon	Weekend	No	No	54	4	Yes
10004	60	Male	64440	9	BBQ Chicken	Medium	35.86	Late Night	Weekend	No	No	40	5	No
10005	25	Female	51655	5	Cheese	Medium	22.65	Afternoon	Weekday	No	No	58	5	No

---

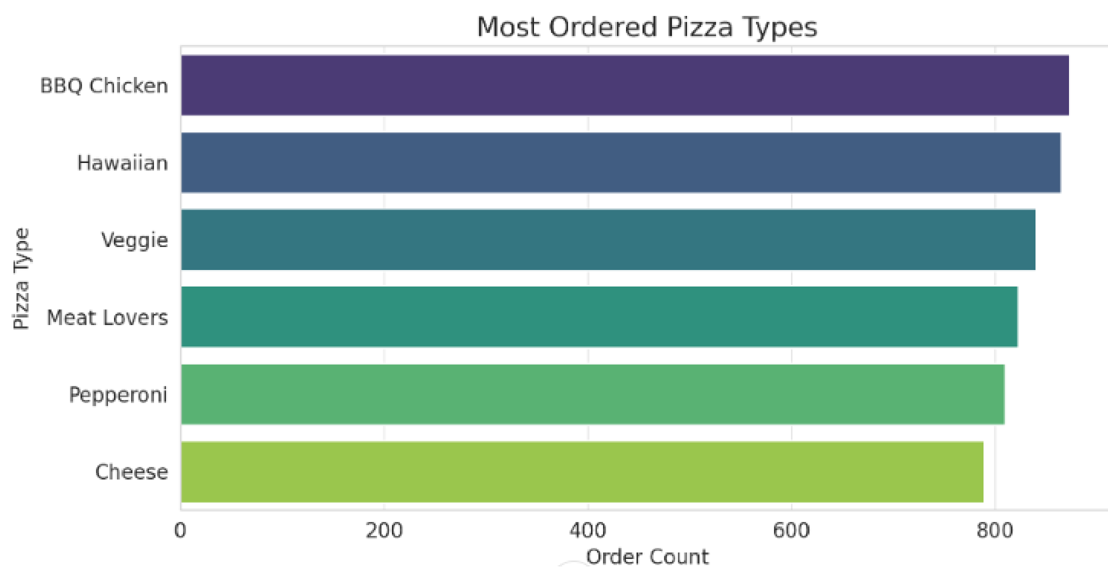
Key Findings from the Summary Statistics

1. **Customer Demographics**
    - Average customer age: **41 years** (ranging from 18 to 64).
    - Income varies widely, with an average of **$59,252** and a range from **$20,005 to $99,991**.
2. **Order Frequency & Spending**
    - Customers order an average of **5 times per month** (ranging from 1 to 9 times).
    - Average spending per order is **$23.72**, with a minimum of **$8** and a maximum of **$39.99**.
3. **Delivery Time & Customer Rating**

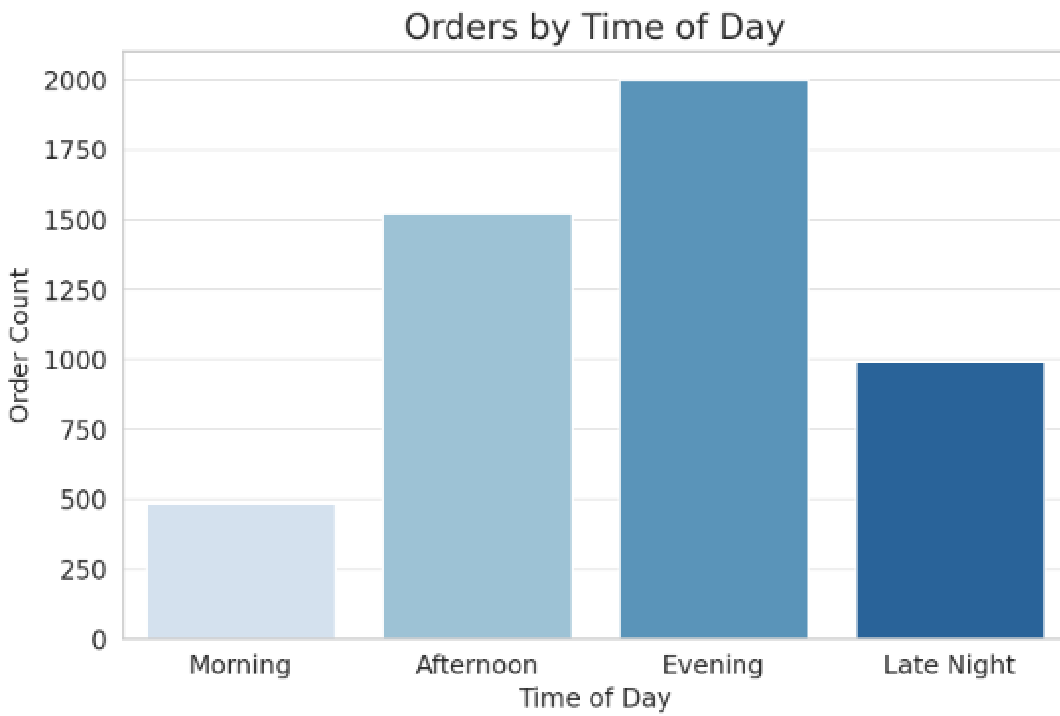
- Average delivery time is **37.2 minutes**, ranging from **15 to 59 minutes**.
- Customer ratings average **3.0** out of 5, with a standard deviation of **1.41**, meaning there is a wide variation in satisfaction.

4. **Categorical Data Insights**
   - **3 gender categories** (Male, Female, Non-binary).
   - **6 pizza types**, with varied customer preferences.
   - **4 time slots** for ordering (Morning, Afternoon, Evening, Late Night).
   - **4 pizza sizes** (Small, Medium, Large, Extra Large).
   - **Weekday vs. Weekend ordering pattern**.
   - **Loyalty program participation** and **discount usage behavior**.
   - **Repeat customer trends** (Yes/No).

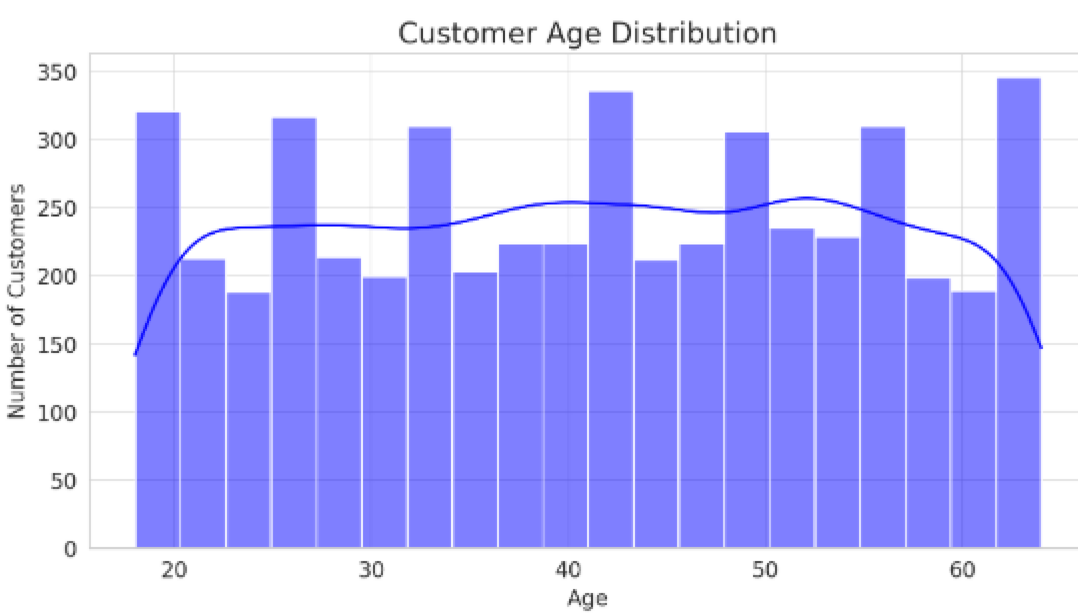
MOST ORDERED PIZZA TYPES



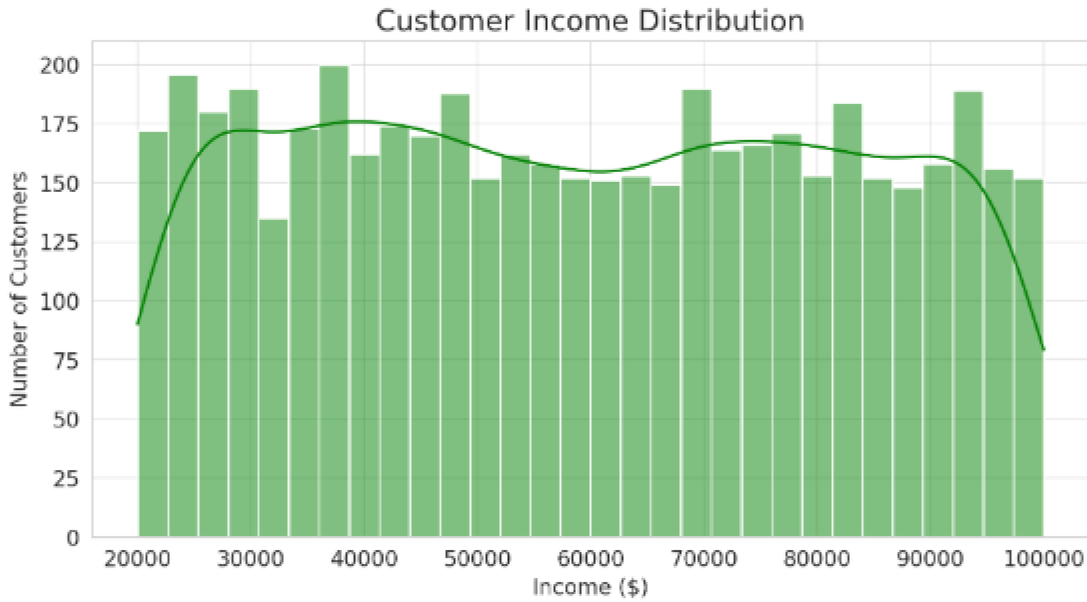
ORDERS BY TIME OF DAY



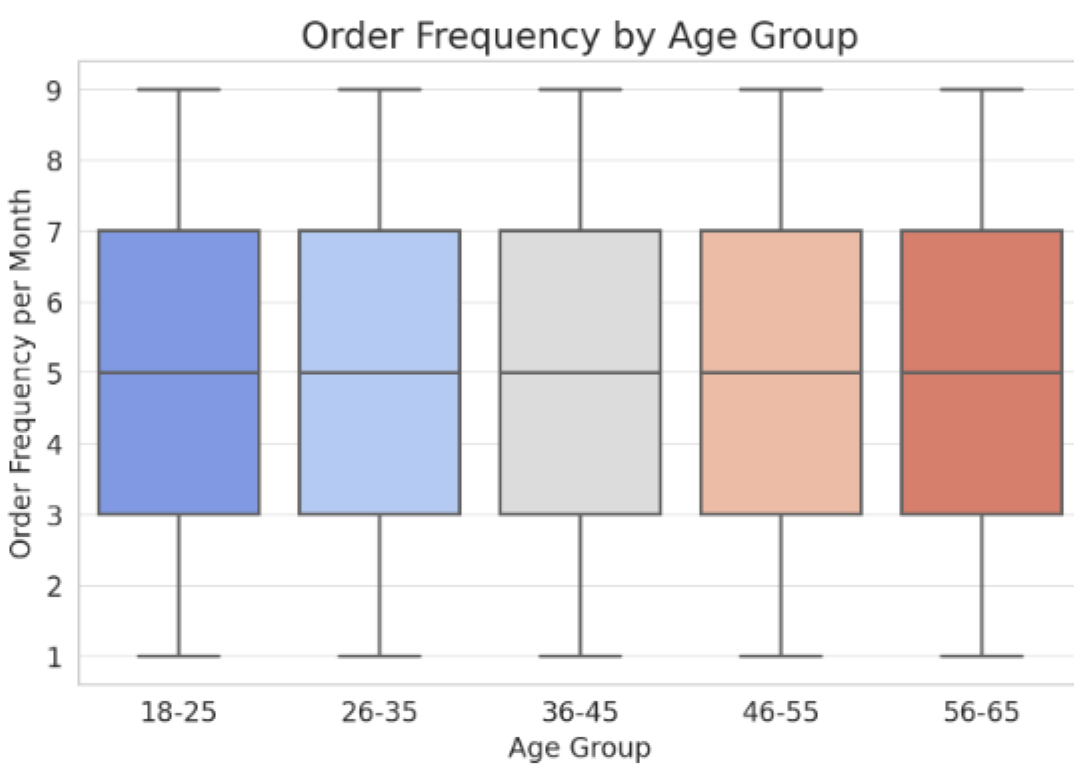
CUSTOMER AGE DISTRIBUTION



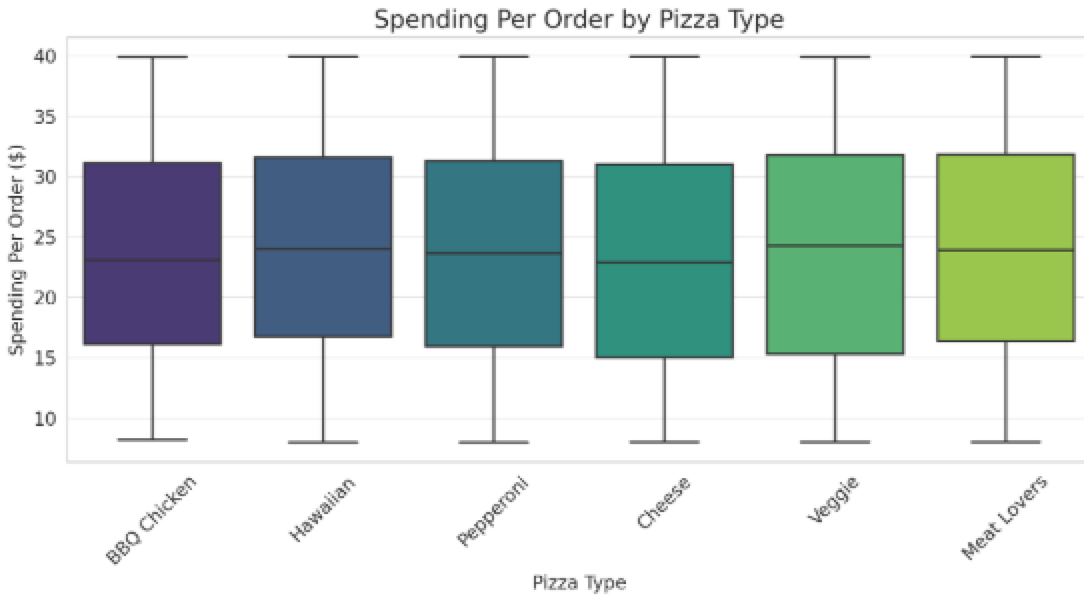
CUSTOMER INCOME DISTRIBUTION



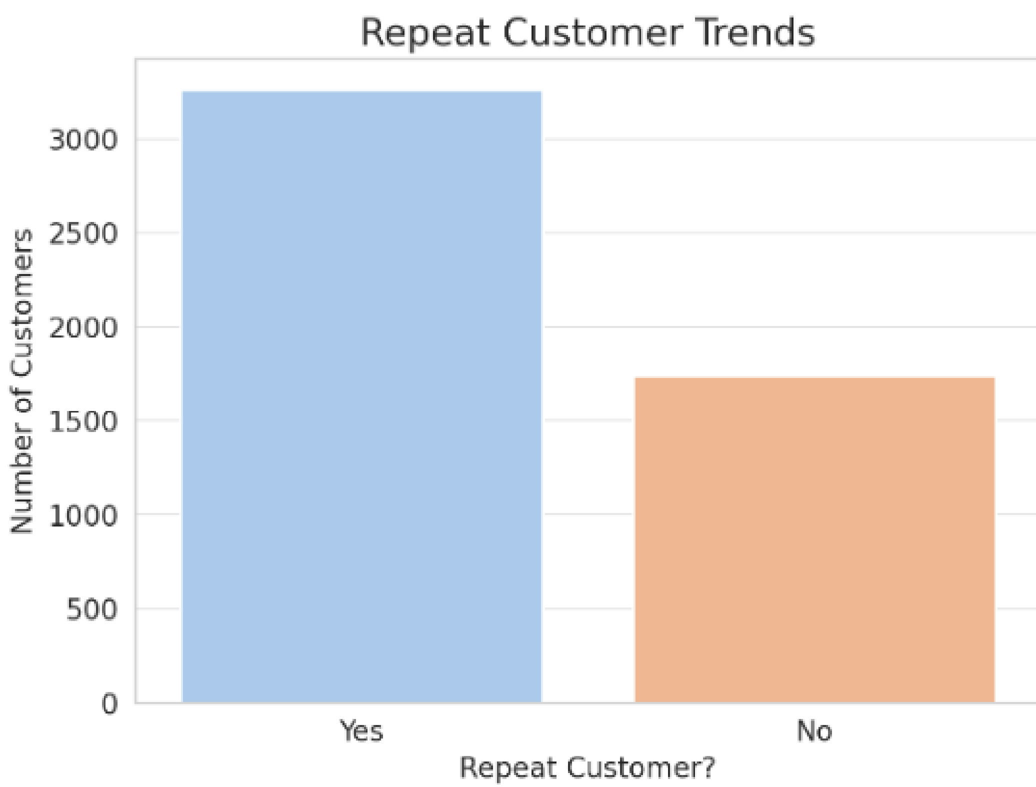
ORDER FREQUENCY BY AGE GROUP



SPENDING PER ORDER BY PIZZA TYPE



REPEAT CUSTOMER TRENDS



Python Script: Data Preprocessing for Pizza Orders data pre-processing

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder


# Load dataset

file_path = "pizza_orders.csv"  # Update this with the correct file path

data = pd.read_csv(file_path)


# Step 1: Handle Missing Values

data.dropna(inplace=True)  # Drop rows with missing values


# Step 2: Encode Categorical Variables

categorical_columns = ['Gender', 'Preferred_Pizza_Type', 'Size', 'Time_of_Order',

                'Day_of_Week', 'Loyalty_Member', 'Discount_Used', 'Repeat_Customer']


label_encoders = {}

for col in categorical_columns:

   le = LabelEncoder()

   data[col] = le.fit_transform(data[col])

   label_encoders[col] = le  # Store encoders for future use
```

```python
# Step 3: Scale Numerical Features

numerical_columns = ['Age', 'Income', 'Order_Frequency', 'Spending_Per_Order',
'Delivery_Time_Minutes']

scaler = StandardScaler()

data[numerical_columns] = scaler.fit_transform(data[numerical_columns])


# Step 4: Remove Outliers (Using Z-score method)

from scipy.stats import zscore

z_scores = np.abs(zscore(data[numerical_columns]))

data = data[(z_scores < 3).all(axis=1)]  # Keep only rows with z-score < 3


# Step 5: Split Data into Training & Testing Sets

X = data.drop(columns=['Repeat_Customer'])  # Features

y = data['Repeat_Customer']  # Target variable


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Save processed data

X_train.to_csv("X_train.csv", index=False)

X_test.to_csv("X_test.csv", index=False)

y_train.to_csv("y_train.csv", index=False)

y_test.to_csv("y_test.csv", index=False)


print(" Data Preprocessing Completed Successfully!")
```

```
Data Preprocessing Completed Successfully!
```

APPLYING LOGISTIC REGRESSION

```
# logistic regression


import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report

import joblib


# Load dataset

file_path = "X_train.csv"  # Update with the correct file path

data = pd.read_csv(file_path)


# If 'Age_Group' column doesn't exist, create it and fill with 0s

if 'Age_Group' not in data.columns:

   data['Age_Group'] = 0  # Assuming 'No' as default if 'Age_Group' is not present


# Identify categorical and numerical columns

categorical_columns = ['Gender', 'Preferred_Pizza_Type', 'Size', 'Time_of_Order',

                  'Day_of_Week', 'Loyalty_Member', 'Discount_Used']

numerical_columns = ['Age', 'Income', 'Order_Frequency', 'Spending_Per_Order',
'Delivery_Time_Minutes']


# One-Hot Encoding for categorical variables
```

```python
data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)

# Scale numerical features
scaler = StandardScaler()
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])

# Convert target variable (Repeat_Customer) to binary (1 for Yes, 0 for No)
# Check if 'Age_Group' has 'Yes' and 'No' values before mapping
if data['Age_Group'].isin(['Yes', 'No']).any():
    data['Age_Group'] = data['Age_Group'].map({'Yes': 1, 'No': 0})

# Drop any remaining NaN values (if any)
data.dropna(inplace=True)

# Define features and target
X = data.drop(columns=['Age_Group'])  # Features
y = data['Age_Group']  # Target

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

# Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predictions and evaluation
```

y_pred = model.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")

print("Classification Report:\n", classification_report(y_test, y_pred))

 # Save the trained model

joblib.dump(model, "logistic_regression_model.pkl")

print("✅ Model saved as logistic_regression_model.pkl")

output

```
Accuracy: 0.90
Classification Report:
               precision    recall  f1-score   support

       18-25       0.87      0.80      0.84       102
       26-35       0.86      0.88      0.87       159
       36-45       0.93      0.93      0.93       162
       46-55       0.90      0.92      0.91       200
       56-65       0.93      0.93      0.93       177

    accuracy                           0.90       800
   macro avg       0.90      0.89      0.89       800
weighted avg       0.90      0.90      0.90       800

✅ Model saved as logistic_regression_model.pkl
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

APPLYING RANDOM FOREST TO DATA SETS

```
# random forest

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib

# Load dataset
file_path = "X_train.csv"  # Update with the correct file path
data = pd.read_csv(file_path)

# Drop rows with missing Age values
data = data.dropna(subset=['Age'])

# Define Age Groups
age_bins = [18, 25, 35, 45, 55, 65]
age_labels = ['18-25', '26-35', '36-45', '46-55', '56-65']
data['Age_Group'] = pd.cut(data['Age'], bins=age_bins, labels=age_labels, right=False)

# Drop the original Age column since we now use Age_Group
data = data.drop(columns=['Age'])
```

```
# Encode Age_Group as a numeric target variable
label_encoder = LabelEncoder()
data['Age_Group'] = label_encoder.fit_transform(data['Age_Group'])

# Identify categorical and numerical columns
categorical_columns = ['Gender', 'Preferred_Pizza_Type', 'Size', 'Time_of_Order',
                       'Day_of_Week', 'Loyalty_Member', 'Discount_Used']
numerical_columns = ['Income', 'Order_Frequency', 'Spending_Per_Order', 'Delivery_Time_Minutes']

# One-Hot Encoding for categorical variables
data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)

# Scale numerical features
scaler = StandardScaler()
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])

# Drop any remaining NaN values (if any)
data.dropna(inplace=True)

# Define features and target
X = data.drop(columns=['Age_Group'])  # Features
y = data['Age_Group']  # Target (encoded)

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

```
# Train Random Forest model

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)


# Predictions and evaluation

y_pred = model.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")


# Save the trained model

joblib.dump(model, "random_forest_age_group_model.pkl")

print("✅ Model saved as random_forest_age_group_model.pkl")
```

OUTPUT:

```
Accuracy: 1.00
✅ Model saved as random_forest_age_group_model.pkl
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3200, 22)
(800, 22)
(3200,)
(800,)
```