**Purchase Pulse Analytics**

🎯 Objective:

To analyze customer purchase behavior using transactional data by:

Understanding purchase frequency

Calculating average purchase value per customer

Identifying buying trends for better marketing and customer segmentation

---

🛠 Technologies Used:

Python Libraries:

pandas for data handling

matplotlib.pyplot for visualizations

---

📂 Dataset Overview:

The dataset purchase\_data.csv contains the following columns:

customer\_id: Unique ID for each customer

product\_id: Product identifier

quantity: Number of items purchased

price: Price per item

purchase\_date: Date of purchase

region: Geographical region of the customer

Sample Records:

customer\_id product\_id quantity price purchase\_date region

1 101 2 10.0 2023-10-01 North

2 102 1 15.0 2023-10-02 South

3 104 4 10.0 2023-10-13 East

---

📊 Step-by-Step Analysis:

✅ 1. Load Data

Loads the dataset from CSV.

Validates file path and error handling built in.

✅ 2. Explore Data

Displays the first 5 rows.

Shows:

Statistical Summary: Mean, std, min, max for numeric fields.

Missing Values Check: Detects and counts NaN values.

✅ 3. Purchase Frequency Analysis

Uses value\_counts() on customer\_id to count how many purchases each customer made.

Visualization: Histogram showing how many customers fall into different frequency bands.

Helps identify loyal vs. one-time customers.

🔍 Example Insight:

A few customers made 4–5 purchases, most only once or twice.

Useful for loyalty segmentation and targeted offers.

✅ 4. Average Purchase Value Analysis

Adds a new column:

purchase\_value = quantity \* price

Aggregates average purchase value per customer using groupby.

Visualization: Histogram of average spend per customer.

Helps identify high-value customers.

🔍 Example Insight:

Some customers spend over ₹100 on average; others spend under ₹20.

Ideal for setting up tiered loyalty programs.

---

📈 Visualization Summary:

1. Histogram: Purchase Frequency

X-axis: Number of purchases

Y-axis: Number of customers

2. Histogram: Average Purchase Value

X-axis: Avg ₹ spent per purchase

Y-axis: Number of customers

Both plots provide a clear distribution and support behavioral segmentation.

---

📌 Conclusions & Business Value:

High-frequency customers should be targeted for retention and reward programs.

High average spenders can be offered premium products or early access offers.

Region-based trends (optional) can also be explored further for location-based marketing.

---

import pandas as pd

import matplotlib.pyplot as plt

def load\_data(file\_path):

"""Load the dataset from a CSV file."""

try:

data = pd.read\_csv(file\_path)

print("Dataset loaded successfully!")

return data

except Exception as e:

print(f"Error loading dataset: {e}")

return None

def explore\_data(data):

"""Perform basic data exploration."""

print("\n--- Dataset Overview ---")

print(data.head()) # Display the first 5 rows

print("\n--- Dataset Summary ---")

print(data.describe()) # Statistical summary

print("\n--- Missing Values ---")

print(data.isnull().sum()) # Check for missing values

def analyze\_purchase\_frequency(data):

"""Analyze purchase frequency by customer."""

print("\n--- Purchase Frequency Analysis ---")

purchase\_frequency = data['customer\_id'].value\_counts()

print(purchase\_frequency.describe()) # Summary of purchase frequency

# Plot purchase frequency distribution

plt.figure(figsize=(10, 6))

purchase\_frequency.hist(bins=50, color='skyblue', edgecolor='black')

plt.title('Purchase Frequency Distribution')

plt.xlabel('Number of Purchases')

plt.ylabel('Number of Customers')

plt.grid(axis='y', alpha=0.75)

plt.show()

def analyze\_average\_purchase\_value(data):

"""Calculate and analyze average purchase value by customer."""

print("\n--- Average Purchase Value Analysis ---")

# Calculate purchase value for each transaction

data['purchase\_value'] = data['quantity'] \* data['price']

# Calculate average purchase value per customer

avg\_purchase\_value = data.groupby('customer\_id')['purchase\_value'].mean()

print(avg\_purchase\_value.describe()) # Summary of average purchase value

# Plot average purchase value distribution

plt.figure(figsize=(10, 6))

avg\_purchase\_value.hist(bins=50, color='lightgreen', edgecolor='black')

plt.title('Average Purchase Value Distribution')

plt.xlabel('Average Purchase Value')

plt.ylabel('Number of Customers')

plt.grid(axis='y', alpha=0.75)

plt.show()

def main():

# File path to the dataset

file\_path = 'purchase\_data.csv'

# Step 1: Load the dataset

data = load\_data(file\_path)

if data is None:

return # Exit if data loading fails

# Step 2: Explore the dataset

explore\_data(data)

# Step 3: Analyze purchase frequency

analyze\_purchase\_frequency(data)

# Step 4: Analyze average purchase value

analyze\_average\_purchase\_value(data)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Data set

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **customer\_id** | **product\_id** | **quantity** | **price** | **purchase\_date** | **region** |
| 1 | 101 | 2 | 10.0 | 2023-10-01 | North |
| 2 | 102 | 1 | 15.0 | 2023-10-02 | South |
| 1 | 103 | 3 | 20.0 | 2023-10-03 | North |
| 3 | 101 | 5 | 5.0 | 2023-10-04 | East |
| 2 | 104 | 2 | 10.0 | 2023-10-05 | South |
| 4 | 105 | 1 | 50.0 | 2023-10-06 | West |
| 5 | 102 | 4 | 15.0 | 2023-10-07 | North |
| 3 | 103 | 2 | 20.0 | 2023-10-08 | East |
| 1 | 104 | 3 | 10.0 | 2023-10-09 | North |
| 2 | 105 | 1 | 50.0 | 2023-10-10 | South |
| 4 | 101 | 2 | 5.0 | 2023-10-11 | West |
| 5 | 103 | 1 | 20.0 | 2023-10-12 | North |
| 3 | 104 | 4 | 10.0 | 2023-10-13 | East |
| 1 | 105 | 2 | 50.0 | 2023-10-14 | North |
| 2 | 101 | 3 | 5.0 | 2023-10-15 | South |

Complete Source Code

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.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

import urllib.request

import uuid

import os

# Step 1: Load and Preprocess the Dataset

# Load dataset

df = pd.read\_csv("online\_shoppers\_intention.csv")

# Handle missing values (if any)

df = df.dropna()

# Encode categorical variables

le = LabelEncoder()

df['VisitorType'] = le.fit\_transform(df['VisitorType'])

df['Month'] = le.fit\_transform(df['Month'])

df['Weekend'] = df['Weekend'].astype(int)

df['Revenue'] = df['Revenue'].astype(int) # Target variable (0 or 1)

# Step 2: Exploratory Data Analysis (EDA)

# Summary statistics

print("Dataset Summary:")

print(df.describe())

# Correlation matrix

plt.figure(figsize=(12, 8))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Matrix of Features')

plt.savefig('correlation\_matrix.png')

plt.close()

# Distribution of ProductRelated\_Duration

plt.figure(figsize=(8, 6))

sns.histplot(df['ProductRelated\_Duration'], bins=30, kde=True)

plt.title('Distribution of Product-Related Page Duration')

plt.xlabel('Duration (seconds)')

plt.ylabel('Count')

plt.savefig('product\_duration\_distribution.png')

plt.close()

# Purchase rate by VisitorType

plt.figure(figsize=(8, 6))

sns.countplot(x='VisitorType', hue='Revenue', data=df)

plt.title('Purchase Rate by Visitor Type')

plt.xlabel('Visitor Type (Encoded)')

plt.ylabel('Count')

plt.savefig('purchase\_by\_visitor\_type.png')

plt.close()

# Step 3: Data Preprocessing

# Features and target

features = ['Administrative', 'Administrative\_Duration', 'Informational', 'Informational\_Duration',

'ProductRelated', 'ProductRelated\_Duration', 'BounceRates', 'ExitRates', 'PageValues',

'SpecialDay', 'Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType',

'VisitorType', 'Weekend']

X = df[features]

y = df['Revenue']

# 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)

# Step 4: Model Training

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Step 5: Model Evaluation

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

print("\nModel Performance:")

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

# Step 6: Save Predictions to CSV

# Create a DataFrame with test features, actual labels, and predictions

results\_df = X\_test.copy()

results\_df['Actual\_Revenue'] = y\_test

results\_df['Predicted\_Revenue'] = y\_pred

# Path to CSV file

csv\_file = 'prediction\_results.csv'

# Check if file exists to append or create new

if os.path.exists(csv\_file):

# Append to existing CSV without header

results\_df.to\_csv(csv\_file, mode='a', header=False, index=False)

else:

# Create new CSV with header

results\_df.to\_csv(csv\_file, mode='w', header=True, index=False)

print(f"Predictions saved to {csv\_file}")

# Step 7: Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.savefig('confusion\_matrix.png')

plt.close()

# Step 8: Feature Importance

feature\_importance = pd.DataFrame({

'Feature': features,

'Importance': model.feature\_importances\_

}).sort\_values('Importance', ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(x='Importance', y='Feature', data=feature\_importance)

plt.title('Feature Importance in Random Forest Model')

plt.savefig('feature\_importance.png')

plt.close()

# Step 9: Save Results Summary

results = f"""

Purchase Pulse Analytics Results

==============================

Dataset: Online Shoppers Purchasing Intention (UCI)

Dataset Size: {len(df)} records

Model: Random Forest Classifier

Accuracy: {accuracy:.2f}

Precision: {precision:.2f}

Recall: {recall:.2f}

==============================

Visualizations Generated:

- correlation\_matrix.png

- product\_duration\_distribution.png

- purchase\_by\_visitor\_type.png

- confusion\_matrix.png

- feature\_importance.png

==============================

Predictions Saved:

- prediction\_results.csv

"""

with open('results\_summary.txt', 'w') as f:

f.write(results)

print("\nResults, visualizations, and predictions saved successfully.")