Text

Description automatically generated

**Activity based**

**Project Report on**

**DWDM**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**By**

**Arun Adhikari**

**SRN No : 202100406**

**Roll No : 38**

**Div : G**

**Third Year Engineering**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

**2023-2024**

**Problem Statement**

**Naive Bayes**

**Business Requirements**

**Stakeholders: Executive leadership, customers,**

**Data Sources: Customer database, content database**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, classification\_report

# Load and preprocess data (replace 'file\_path' with the actual path of the Excel file)

file\_path = 'Online\_Retail.xlsx'

data = pd.read\_excel(file\_path, usecols=["InvoiceNo", "CustomerID", "Quantity", "UnitPrice", "InvoiceDate"])

data = data.dropna(subset=['CustomerID'])

# Calculate 'TotalSpent' for each transaction

data['TotalSpent'] = data['Quantity'] \* data['UnitPrice']

# Aggregate per CustomerID to create features for classification

customer\_data = data.groupby('CustomerID').agg({

'TotalSpent': 'sum',

'InvoiceNo': 'nunique'

}).rename(columns={'InvoiceNo': 'PurchaseFrequency'})

# Create classification target based on spending behavior

# We'll define 'TotalSpent' classes: low, medium, high spenders

customer\_data['SpenderCategory'] = pd.qcut(customer\_data['TotalSpent'], q=3, labels=['Low', 'Medium', 'High'])

# Define features and target

X = customer\_data[['TotalSpent', 'PurchaseFrequency']]

y = customer\_data['SpenderCategory']

# Standardize the features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.3, random\_state=42)

# Initialize and train the Naive Bayes classifier

nb\_model = GaussianNB()

nb\_model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = nb\_model.predict(X\_test)

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Assuming y\_test and y\_pred are defined from the Naive Bayes code

conf\_matrix = confusion\_matrix(y\_test, y\_pred, labels=['Low', 'Medium', 'High'])

# Plot the confusion matrix

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=['Low', 'Medium', 'High'],

yticklabels=['Low', 'Medium', 'High'])

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix for Naive Bayes Classification")

plt.show()

A screenshot of a graph

Description automatically generated

