EX.NO: 1	Customer Segmentation Analysis

OBJECTIVE:

To classify customers into distinct groups based on purchasing behavior and demographics, enabling targeted marketing strategies and personalized service offerings.

READING MATERIAL:

Customer segmentation analysis involves dividing a company's customer base into distinct segments or groups based on shared characteristics. These characteristics can be demographic (age, gender, income level), geographic (location, climate), psychographic (lifestyle, values, interests), or behavioral (purchase history, brand loyalty, product usage). The main objectives are to understand the specific needs and preferences of each segment, create personalized marketing campaigns, and improve product offerings to better meet customer needs.

For example, a clothing retailer might segment its customers into groups such as young professionals, parents, and teenagers. By understanding the preferences of each group, the retailer can tailor its product lines, advertising messages, and promotional offers to resonate with each segment, thereby increasing sales and customer satisfaction.

Segmentation can be done using various techniques, such as:

- Clustering Algorithms: Like K-means or hierarchical clustering, which group customers based on similarities in data.
- Factor Analysis: Which reduces data dimensions and identifies underlying factors that differentiate customers.
- Decision Trees: Which classify customers based on decision rules derived from their attributes.

EX.NO: 1	Customer Segmentation Analysis

OBJECTIVE

To classify customers into distinct groups based on purchasing behavior and demographics, enabling targeted marketing strategies and personalized service offerings.

AIM

To categorize customers into groups for targeted marketing and improved service.

ALGORITHM / PROCEDURE

Step 1. Importing Libraries.
Step 2. Exploration of data.
Step 3. Data Visualization.
Step 4. Clustering using K-Means.
Step 5. Selection of Clusters.
Step 6. Ploting the Cluster Boundry and Clusters.
Step 7. 3D Plot of Clusters.

PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as py
import plotly.graph_objs as go

from sklearn.cluster import KMeans
import warnings
import os

warnings.filterwarnings("ignore")
py.offline.init_notebook_mode(connected=True)

# Load the dataset
```

```
df = pd.read_csv('Mall_Customers.csv')

# Exploratory Data Analysis
print("Data Head:")
print(df.head())
print("\nData Shape:")
print(df.shape)
print("\nData Description:")
print(df.describe())
print("\nData Types:")
print(df.dtypes)
print("\nNull Values:")
print(df.isnull().sum())

# Visualizations
plt.style.use('fivethirtyeight')

# Distribution plots
plt.figure(1, figsize=(15, 6))
n = 0
for x in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.distplot(df[x], bins=20)
    plt.title('Distplot of {}'.format(x))
plt.show()

# Gender count plot
plt.figure(1, figsize=(15, 5))
sns.countplot(y='Gender', data=df)
plt.show()

# Regression plots
plt.figure(1, figsize=(15, 7))
n = 0
for x in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    for y in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
        n += 1
        plt.subplot(3, 3, n)
        plt.subplots_adjust(hspace=0.5, wspace=0.5)
        sns.regplot(x=x, y=y, data=df)
        plt.ylabel(y.split()[0] + ' ' + y.split()[1] if len(y.split()) > 1
else y)
plt.show()
```

```
# Scatter plot of Age vs. Annual Income with respect to Gender
plt.figure(1, figsize=(15, 6))
for gender in ['Male', 'Female']:
    plt.scatter(x='Age', y='Annual Income (k$)', data=df[df['Gender'] ==
gender],
                s=200, alpha=0.5, label=gender)
plt.xlabel('Age')
plt.ylabel('Annual Income (k$)')
plt.title('Age vs Annual Income w.r.t Gender')
plt.legend()
plt.show()

# Scatter plot of Annual Income vs. Spending Score with respect to Gender
plt.figure(1, figsize=(15, 6))
for gender in ['Male', 'Female']:
    plt.scatter(x='Annual Income (k$)', y='Spending Score (1-100)',
                data=df[df['Gender'] == gender], s=200, alpha=0.5,
label=gender)
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Annual Income vs Spending Score w.r.t Gender')
plt.legend()
plt.show()

# Violin and Swarm plots for Age, Annual Income, and Spending Score by
Gender
plt.figure(1, figsize=(15, 7))
n = 0
for cols in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.violinplot(x=cols, y='Gender', data=df, palette='vlag')
    sns.swarmplot(x=cols, y='Gender', data=df)
    plt.ylabel('Gender' if n == 1 else '')
    plt.title('Boxplots & Swarmplots' if n == 2 else '')
plt.show()

# K-Means Clustering for Age and Spending Score
X1 = df[['Age', 'Spending Score (1-100)']].iloc[:, :].values
inertia = []
for n in range(1, 11):
    algorithm = KMeans(n_clusters=n, init='k-means++', n_init=10,
max_iter=300,
```

```
                        tol=0.0001, random_state=111, algorithm='elkan')
    algorithm.fit(X1)
    inertia.append(algorithm.inertia_)

plt.figure(1, figsize=(15, 6)) plt.plot(np.arange(1,
11), inertia, 'o') plt.plot(np.arange(1, 11),
inertia, '-', alpha=0.5) plt.xlabel('Number of
Clusters') plt.ylabel('Inertia')
plt.show()

algorithm = KMeans(n_clusters=4, init='k-means++', n_init=10,
max_iter=300,
                   tol=0.0001, random_state=111, algorithm='elkan')
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_

h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() + 1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))
Z = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])

plt.figure(1, figsize=(15, 7))
plt.clf()
Z = Z.reshape(xx.shape)
plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Pastel2, aspect='auto', origin='lower')
plt.scatter(x='Age', y='Spending Score (1-100)', data=df, c=labels1,
s=200)
plt.scatter(x=centroids1[:, 0], y=centroids1[:, 1], s=300, c='red',
alpha=0.5)
plt.ylabel('Spending Score (1-100)')
plt.xlabel('Age')
plt.show()

# K-Means Clustering for Annual Income and Spending Score
X2 = df[['Annual Income (k$)', 'Spending Score (1-100)']].iloc[:,
:].values
inertia = []
for n in range(1, 11):
```

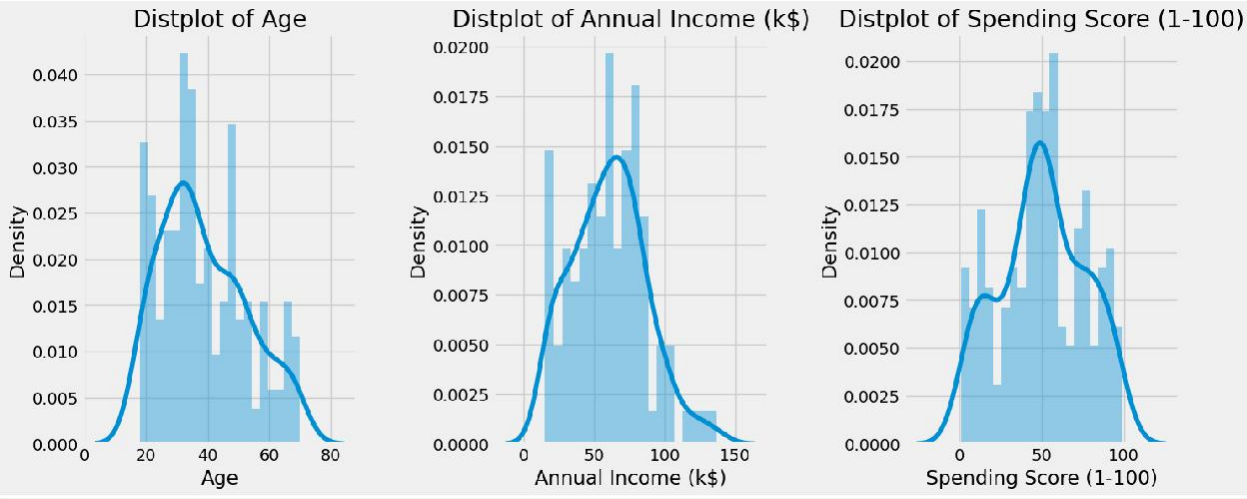
```
    algorithm = KMeans(n_clusters=n, init='k-means++', n_init=10,
max_iter=300,
                       tol=0.0001, random_state=111, algorithm='elkan')
    algorithm.fit(X2)
    inertia.append(algorithm.inertia_)

plt.figure(1, figsize=(15, 6)) plt.plot(np.arange(1,
11), inertia, 'o') plt.plot(np.arange(1, 11),
inertia, '-', alpha=0.5) plt.xlabel('Number of
Clusters') plt.ylabel('Inertia')
plt.show()

algorithm = KMeans(n_clusters=5, init='k-means++', n_init=10,
max_iter=300,
                   tol=0.0001, random_state=111, algorithm='elkan')
algorithm.fit(X2)
labels2 = algorithm.labels_
centroids2 = algorithm.cluster_centers_

h = 0.02
x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() + 1
y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))
Z2 = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])

plt.figure(1, figsize=(15, 7))
plt.clf()
Z2 = Z2.reshape(xx.shape) plt.imshow(Z2,
interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Pastel2, aspect='auto', origin='lower')
plt.scatter(x='Annual Income (k$)', y='Spending Score (1-100)', data=df,
c=labels2, s=200)
plt.scatter(x=centroids2[:, 0], y=centroids2[:, 1], s=300, c='red',
alpha=0.5)
plt.ylabel('Spending Score (1-100)')
plt.xlabel('Annual Income (k$)')
plt.show()

# K-Means Clustering for Age, Annual Income, and Spending Score
X3 = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].iloc[:,
:].values
inertia = []
```

```
for n in range(1, 11):
    algorithm = KMeans(n_clusters=n, init='k-means++', n_init=10,
max_iter=300,
                       tol=0.0001, random_state=111, algorithm='elkan')
    algorithm.fit(X3)
    inertia.append(algorithm.inertia_)

plt.figure(1, figsize=(15, 6)) plt.plot(np.arange(1,
11), inertia, 'o') plt.plot(np.arange(1, 11),
inertia, '-', alpha=0.5) plt.xlabel('Number of
Clusters') plt.ylabel('Inertia')
plt.show()

algorithm = KMeans(n_clusters=6, init='k-means++', n_init=10,
max_iter=300,
                   tol=0.0001, random_state=111, algorithm='elkan')
algorithm.fit(X3)
labels3 = algorithm.labels_
centroids3 = algorithm.cluster_centers_
df['label3'] = labels3

trace1 =
    go.Scatter3d( x=df['Age'],
    y=df['Spending Score (1-100)'],
    z=df['Annual Income (k$)'],
    mode='markers',
    marker=dict(
        color=df['label3'],
        size=10,  # Adjusted for better visualization
        line=dict(
            color=df['label3'],
            width=0.5  # Adjusted for better visualization
        ),
        opacity=0.8
    )
)

data = [trace1]
layout = go.Layout(
    margin=dict(
         l=0,
        r=0, b=0,
        t=0
```

```
    ),
    title='Clusters',
    scene=dict(
        xaxis=dict(title='Age'),
        yaxis=dict(title='Spending Score'),
        zaxis=dict(title='Annual Income (k$)')
    )
)

fig = go.Figure(data=data, layout=layout)
py.offline.iplot(fig
```

OUTPUT

```
Data Head:
   CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40

Data Shape:
(200, 5)

Data Description:
       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
50%    100.500000   36.000000           61.500000               50.000000
75%    150.250000   49.000000           78.000000               73.000000
max    200.000000   70.000000          137.000000               99.000000

Data Types:
CustomerID                 int64
Genre                     object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object

Null Values:
CustomerID                 0
Genre                      0
Age                        0
Annual Income (k$)         0
Spending Score (1-100)     0
dtype: int64
```



Criteria	Marks
Preparation	**/20**
Observation	**/25**
Interpretation of result	**/20**
viva	**/10**
Total	**/75**
Faculty signature with date	

RESULT

Thus, using customer segmentation, targeted marketing campaigns have been implemented successfully.

EX.NO: 2	Churn Prediction

OBJECTIVE

To identify customers who are at risk of leaving the service, allowing for proactive retention strategies and interventions to reduce churn rates.

READING MATERIAL:

Churn prediction is a process used to identify customers who are likely to leave a company or stop using its products or services. This is particularly crucial for subscription-based businesses, such as telecom companies, SaaS providers, or online streaming services, where retaining customers is essential for sustained revenue.

The process involves analyzing customer data to identify patterns or indicators that precede churn. These indicators can include a decrease in usage frequency, negative feedback, complaints, or changes in purchasing behavior. Predictive models, such as logistic regression, decision trees, random forests, or neural networks, are often used to forecast churn probability.

For instance, a telecom company might analyze call and data usage patterns, billing history, and customer service interactions to predict which customers are at risk of switching to a competitor. By identifying these customers early, the company can offer targeted retention campaigns, such as special discounts or personalized offers, to encourage them to stay.

EX.NO: 2	Churn Prediction

OBJECTIVE

To identify customers who are at risk of leaving the service, allowing for proactive retention strategies and interventions to reduce churn rates.

AIM

To predict and prevent customer attrition with targeted retention strategies.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score

print("✅ Libraries imported successfully!")

try:
    df = pd.read_csv('Telco-Customer-Churn.csv')
    print("\n✅ Dataset loaded successfully!")
except FileNotFoundError:
    print("❌ Error: 'Telco-Customer-Churn.csv' not found. Please ensure
the file is in the correct directory.")
    exit()

print("\n--- First 5 Rows of the Dataset ---")
print(df.head())

print("\n--- Dataset Information ---")
df.info()

print("\n--- Statistical Summary ---")
print(df.describe())

df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

print("\n--- Missing Values ---")
print(df.isnull().sum())

print("\n  Generating visualizations...")

plt.style.use('seaborn-v0_8-whitegrid')

plt.figure(figsize=(6, 4))
sns.countplot(x='Churn', data=df)
plt.title('Distribution of Customer Churn')
plt.xlabel('Churn')
plt.ylabel('Number of Customers')
```

```
plt.show()


categorical_features = ['gender', 'SeniorCitizen', 'Partner',
'Dependents', 'Contract']

fig, axes = plt.subplots(nrows=1, ncols=len(categorical_features),
figsize=(20, 5), sharey=True)
fig.suptitle('Churn Rate by Customer Demographics and Contract Type',
fontsize=16)

for i, feature in enumerate(categorical_features):
    sns.countplot(x=feature, hue='Churn', data=df, ax=axes[i])
    axes[i].set_title(f'Churn by {feature}')
    axes[i].set_xlabel('')
    axes[i].legend(title='Churn')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

print("✅ Visualizations generated.") print("\n⚙

Starting data preprocessing...")

median_total_charges = df['TotalCharges'].median()
df['TotalCharges'].fillna(median_total_charges, inplace=True)
print(f"✅ Missing 'TotalCharges' filled with median value:
{median_total_charges}")

df.drop('customerID', axis=1, inplace=True)
print("✅ 'customerID' column dropped.")

df_processed = df.copy()
for column in df_processed.columns:
    if df_processed[column].dtype == 'object' and
len(df_processed[column].unique()) <= 2:
        le = LabelEncoder()
        df_processed[column] = le.fit_transform(df_processed[column])

df_processed = pd.get_dummies(df_processed, drop_first=True)

print("✅ Categorical features encoded.")
print("\n--- Data Head After Preprocessing ---")
print(df_processed.head())
```

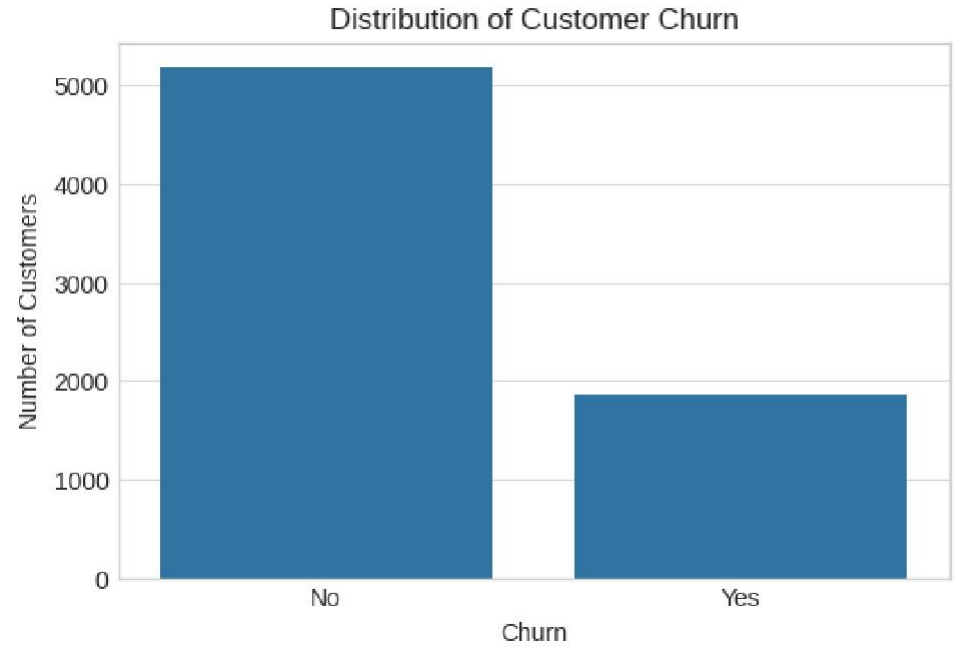
```
print("\n  Splitting data into training and testing sets...")

X = df_processed.drop('Churn', axis=1)
y = df_processed['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42, stratify=y)

scaler = StandardScaler()
numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges']
X_train[numerical_features] =
scaler.fit_transform(X_train[numerical_features])
X_test[numerical_features] = scaler.transform(X_test[numerical_features])

print(f"✅ Data split complete. Training set has {X_train.shape[0]}
samples.")

print("\n□ Selecting and training the model...")

model = LogisticRegression(random_state=42, max_iter=1000)

model.fit(X_train, y_train)

print("✅ Model trained successfully!")

print("\n📈 Evaluating the model...")

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.2%}")
print("\n--- Confusion Matrix ---")
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not
Churn', 'Churn'], yticklabels=['Not Churn', 'Churn'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()
print("\n--- Classification Report ---")
print(classification_report(y_test, y_pred, target_names=['Not Churn',
'Churn']))
print("✅ Evaluation complete. The model is ready to identify at-risk
customers.")
```

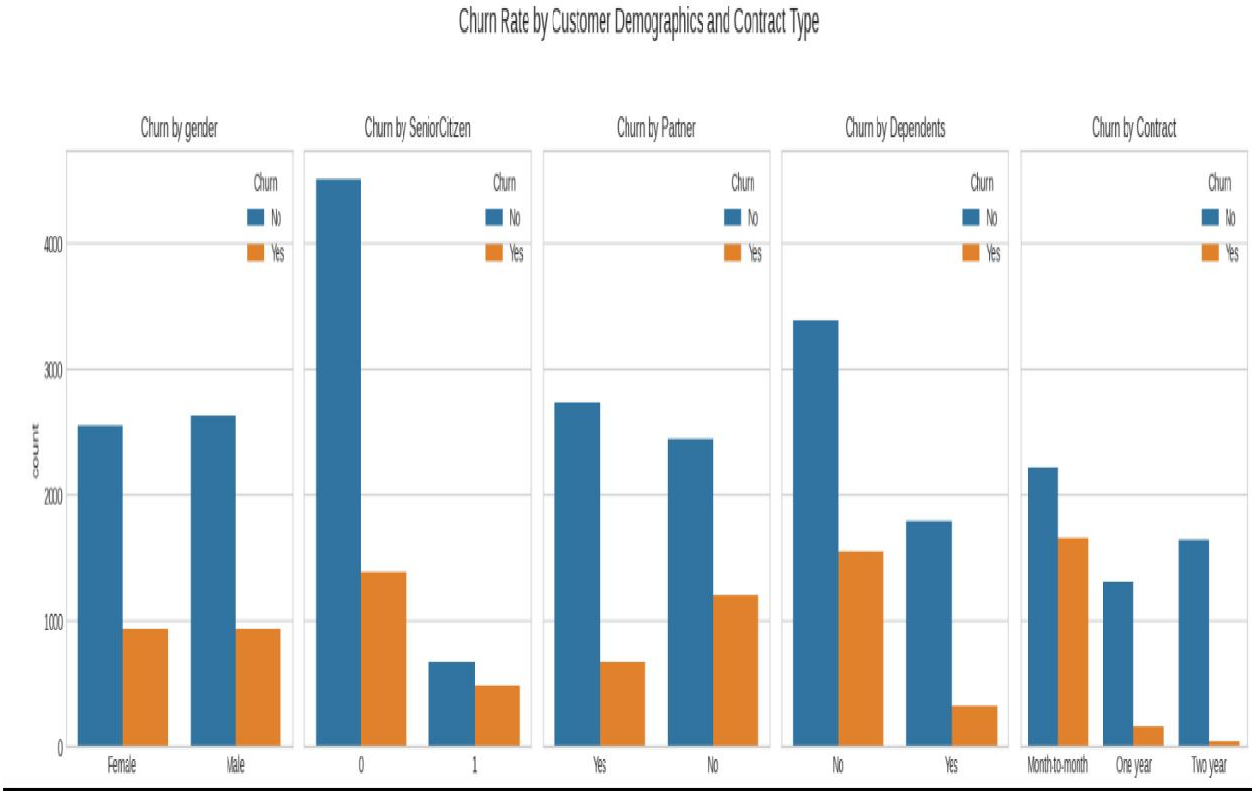
OUTPUT

```
--- First 5 Rows of the Dataset ---
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0  No phone service             DSL             No  ...               No
1                No             DSL            Yes  ...              Yes
2                No             DSL            Yes  ...               No
3  No phone service             DSL            Yes  ...              Yes
4                No     Fiber optic             No  ...               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

                PaymentMethod MonthlyCharges  TotalCharges Churn
0            Electronic check          29.85         29.85    No
1                Mailed check          56.95        1889.5    No
2                Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4            Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

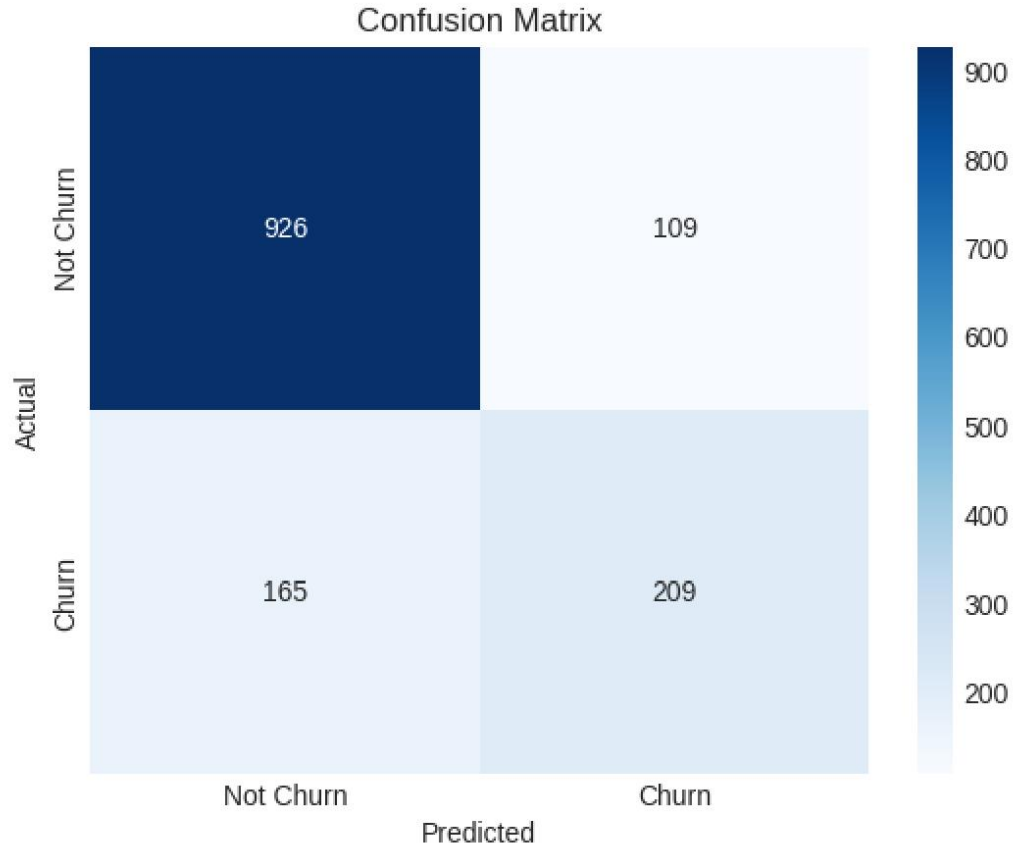
```
--- Dataset Information ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #    Column            Non-Null Count  Dtype
---   ------            --------------  -----
 0    customerID        7043 non-null   object
 1    gender            7043 non-null   object
 2    SeniorCitizen     7043 non-null   int64
 3    Partner           7043 non-null   object
 4    Dependents        7043 non-null   object
 5    tenure            7043 non-null   int64
 6    PhoneService      7043 non-null   object
 7    MultipleLines     7043 non-null   object
 8    InternetService   7043 non-null   object
 9    OnlineSecurity    7043 non-null   object
 10   OnlineBackup      7043 non-null   object
 11   DeviceProtection  7043 non-null   object
 12   TechSupport       7043 non-null   object
 13   StreamingTV       7043 non-null   object
 14   StreamingMovies   7043 non-null   object
 15   Contract          7043 non-null   object
 16   PaperlessBilling  7043 non-null   object
 17   PaymentMethod     7043 non-null   object
 18   MonthlyCharges    7043 non-null   float64
 19   TotalCharges      7043 non-null   object
 20   Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Generating visualizations...





```
df['TotalCharges'].fillna(median_total_charges, inplace=True)
```




```
--- Classification Report ---
              precision    recall  f1-score   support

   Not Churn       0.85      0.89      0.87      1035
       Churn       0.66      0.56      0.60       374

    accuracy                           0.81      1409
   macro avg       0.75      0.73      0.74      1409
weighted avg       0.80      0.81      0.80      1409
```

Criteria	Marks
Preparation	**/20**
Observation	**/25**
Interpretation of result	**/20**
viva	**/10**
Total	**/75**
Faculty signature with date	

RESULT

Thus, using churn prediction models, retention strategies have been implemented successfully, reducing customer attrition.

EX.NO: 3	Sales Forecasting

OBJECTIVE

To predict future sales trends using historical data and market analysis, aiding in inventory management, budgeting, and strategic planning.

READING MATERIAL:

Sales forecasting is the practice of predicting future sales revenue based on historical sales data, market trends, economic conditions, and other relevant factors. Accurate sales forecasts are critical for effective business planning, budgeting, and resource allocation. They help companies manage inventory, plan production, and set realistic sales targets.

There are several methods used in sales forecasting, including:

- Time Series Analysis: Techniques like moving averages, exponential smoothing, and ARIMA models that analyze patterns in historical sales data over time.
- Causal Models: Which consider relationships between sales and other variables, such as advertising spend, economic indicators, or market trends.

Machine Learning Models: Such as decision trees, support vector machines, or neural networks, which can capture complex, non-linear relationships in data.

EX.NO: 3	Sales Forecasting

OBJECTIVE

To predict future sales trends using historical data and market analysis, aiding in inventory management, budgeting, and strategic planning.

AIM

To predict future sales to optimize inventory and resource allocation

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

Step 8: Forecast Future Sales

PROGRAM

```
import warnings
warnings.filterwarnings("ignore")

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.statespace.sarimax import SARIMAX
from datetime import timedelta

CSV_PATH = "sales_data.csv"
DATE_COL = "date"
TARGET_COL = "sales"

def eval_forecast(y_true, y_pred, label="Validation"):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = mean_squared_error(y_true, y_pred)
    mape = (np.abs((y_true - y_pred) / np.maximum(y_true, 1e-9))).mean() *
100
    print(f"{label} MAE : {mae:,.2f}")
    print(f"{label} RMSE: {rmse:,.2f}")
    print(f"{label} MAPE: {mape:,.2f}%")
    return mae, rmse, mape
df = pd.read_csv(CSV_PATH)
if DATE_COL not in df.columns or TARGET_COL not in df.columns:
    raise ValueError("CSV must contain at least 'date' and 'sales'
columns.")

df[DATE_COL] = pd.to_datetime(df[DATE_COL])
df = df.sort_values(DATE_COL).set_index(DATE_COL)

if FREQ is None:
    FREQ = pd.infer_freq(df.index)
    if FREQ is None:
        FREQ = "D"

if FREQ in ["D", "H"]:
    df = df.asfreq(FREQ)
    df[TARGET_COL] = df[TARGET_COL].fillna(0)
else:
```

```
    df = df.resample(FREQ).sum()

numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
drivers = [c for c in numeric_cols if c != TARGET_COL]

val_horizon = min(FORECAST_STEPS, int(0.5 * len(df)))
train = df.iloc[:-val_horizon]
valid = df.iloc[-val_horizon:]

def build_exog(frame, driver_cols):
    if not driver_cols:
        return None, None
    scaler = StandardScaler()
    X =
scaler.fit_transform(frame[driver_cols].fillna(method="ffill").fillna(0.0)
)
    return X, scaler

X_train, scaler = build_exog(train, drivers)
X_valid =
scaler.transform(valid[drivers].fillna(method="ffill").fillna(0.0)) if
drivers else None

y_train = train[TARGET_COL]
y_valid = valid[TARGET_COL]

seasonal_periods_map = {"D": 7, "W": 52, "M": 12}
m = seasonal_periods_map.get(FREQ, 7)

order = (1, 1, 1)
seasonal_order = (1, 1, 1, m)

print("Fitting SARIMAX...")
model = SARIMAX(
    y_train,
    exog=X_train,
    order=order,
    seasonal_order=seasonal_order,
    enforce_stationarity=False,
    enforce_invertibility=False,
)
res = model.fit(disp=False)
pred_valid = res.get_forecast(steps=len(valid), exog=X_valid)
yhat_valid = pred_valid.predicted_mean
ci_valid = pred_valid.conf_int(alpha=0.05)  # 95% CI
```

```
print("\n=== Validation Performance ===")
mae, rmse, mape = eval_forecast(y_valid, yhat_valid, label="Validation")

X_all =
scaler.fit_transform(df[drivers].fillna(method="ffill").fillna(0.0)) if
drivers else None
last_date = df.index[-1]
future_index = pd.date_range(last_date +
pd.tseries.frequencies.to_offset(FREQ), periods=FORECAST_STEPS, freq=FREQ)

future_exog = None
if drivers:
    last_row = df[drivers].iloc[-1:].copy()
    future_exog_plan = pd.DataFrame(np.repeat(last_row.values,
FORECAST_STEPS, axis=0), columns=drivers, index=future_index)
    future_exog =
scaler.transform(future_exog_plan.fillna(method="ffill").fillna(0.0))

print("\nRefitting on full history...")
model_full = SARIMAX(
    df[TARGET_COL],
    exog=X_all,
    order=order,
    seasonal_order=seasonal_order,
    enforce_stationarity=False,
    enforce_invertibility=False,
)
res_full = model_full.fit(disp=False)

pred_future = res_full.get_forecast(steps=FORECAST_STEPS,
exog=future_exog)
yhat_future = pred_future.predicted_mean
ci_future = pred_future.conf_int(alpha=0.05)
forecast_sigma = rmse if np.isfinite(rmse) and rmse > 0 else
(np.std(y_train) * 0.5 + 1e-6)
safety_stock = SERVICE_Z * forecast_sigma * np.sqrt(max(LEAD_TIME, 1))
lt_periods = min(LEAD_TIME, len(yhat_future))
lead_time_demand = yhat_future.iloc[:lt_periods].sum() if lt_periods > 0
else yhat_future.mean() * LEAD_TIME
reorder_point = lead_time_demand + safety_stock

if "unit_cost" in df.columns:
    unit_cost = float(df["unit_cost"].iloc[-1])
else:
```

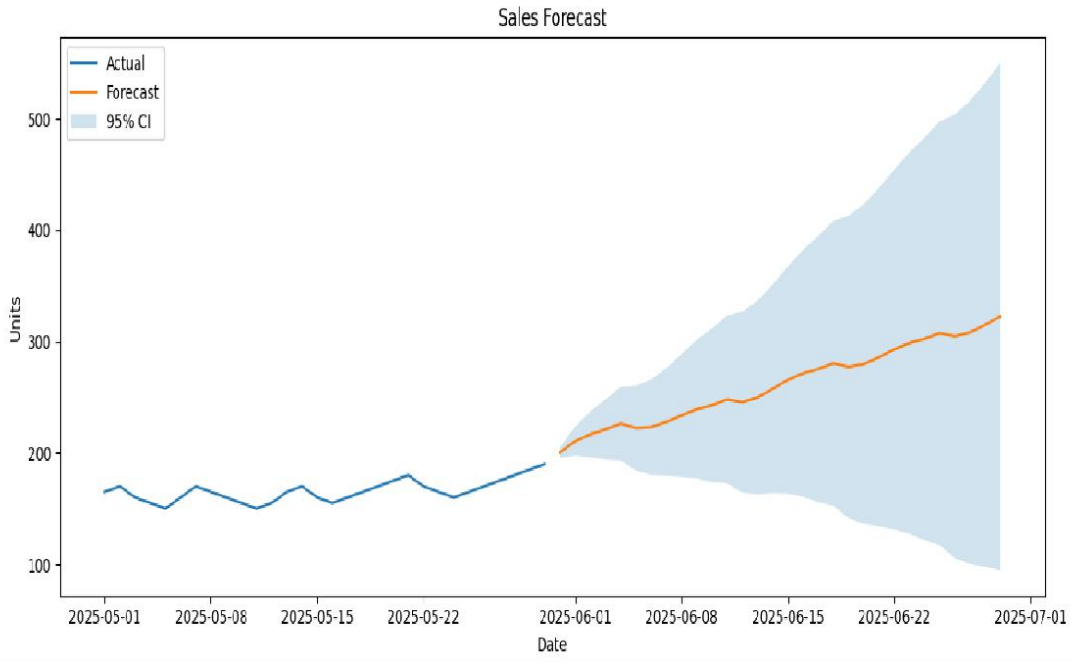
```
    unit_cost = DEFAULT_UNIT_COST

projected_units = yhat_future.sum()
projected_cogs = projected_units * unit_cost

results = pd.DataFrame({
    "date": df.index.tolist() + yhat_future.index.tolist(), "type":
    ["actual"] * len(df) + ["forecast"] * len(yhat_future), "sales":
    pd.concat([df[TARGET_COL], yhat_future]).values
}).set_index("date")

forecast_out =
    pd.DataFrame({ "date":
    yhat_future.index, "forecast":
    yhat_future.values,
    "lo_95": ci_future.iloc[:, 0].values,
    "hi_95": ci_future.iloc[:, 1].values
}).set_index("date")

results.to_csv("sales_forecast_results.csv")
forecast_out.to_csv("sales_forecast_only.csv")

print("\n=== Planning Aids ===")
print(f"Lead time (periods): {LEAD_TIME}")
print(f"Estimated forecast sigma: {forecast_sigma:,.2f}")
print(f"Safety stock (units): {safety_stock:,.0f}")
print(f"Reorder point (units): {reorder_point:,.0f}")
print(f"Projected units next {FORECAST_STEPS} {FREQ}:
{projected_units:,.0f}")
print(f"Projected COGS (@ {unit_cost:.2f} per unit):
{projected_cogs:,.2f}")

try:
    plt.figure(figsize=(12, 5))
    plt.plot(df.index, df[TARGET_COL], label="Actual")
    plt.plot(yhat_future.index, yhat_future, label="Forecast")
    plt.fill_between(yhat_future.index, ci_future.iloc[:,0],
ci_future.iloc[:,1], alpha=0.2, label="95% CI")
    plt.title("Sales Forecast")
    plt.xlabel("Date")
    plt.ylabel("Units")
    plt.legend()
    plt.tight_layout()
    plt.savefig("sales_forecast_plot.png", dpi=140)
    print("\nSaved: sales_forecast_results.csv, sales_forecast_only.csv,
```

```
    sales_forecast_plot.png")
```

```
except Exception as e:
    print("Plotting skipped:", e)
```

OUTPUT:

```
Fitting SARIMAX...

=== Validation Performance ===
Validation MAE : 18.55
Validation RMSE: 396.09
Validation MAPE: 10.77%

Refitting on full history...

=== Planning Aids ===
Lead time (periods): 7
Estimated forecast sigma: 396.09
Safety stock (units): 1,729
Reorder point (units): 3,250
Projected units next 30 D: 7,851
Projected COGS (@ 8.77 per unit): 68,854.82

Saved: sales_forecast_results.csv, sales_forecast_only.csv, sales_forecast_plot.png
```



Criteria	Marks
Preparation	**/20**
Observation	**/25**
Interpretation of result	**/20**
viva	**/10**
Total	**/75**
Faculty signature with date	

RESULT

Thus, using sales forecasting techniques, inventory management has been optimized successfully.

EX.NO: 4	Sentiment Analysis on Social Media Data

OBJECTIVE

To gauge public opinion and sentiment towards the brand or products, providing insights for marketing campaigns and customer service improvements.

READING MATERIAL:

Sentiment analysis, also known as opinion mining, involves analyzing text data from social media platforms, forums, blogs, and other online sources to determine the emotional tone and sentiment expressed by users. It helps businesses understand how people feel about their brand, products, services, or specific events.

The sentiment can be categorized as positive, negative, or neutral, and more advanced analyses might classify emotions such as joy, anger, sadness, or surprise. Natural language processing (NLP) techniques are used to process and analyze the text data, identifying keywords, phrases, and context to assess sentiment.

For example, a company might use sentiment analysis to monitor reactions to a new product launch on platforms like Twitter, Facebook, or Instagram. By analyzing comments and posts, the company can gauge public perception, identify potential issues early, and respond to customer concerns proactively.

EX.NO: 4	Sentiment Analysis on Social Media Data

OBJECTIVE

To gauge public opinion and sentiment towards the brand or products, providing insights for marketing campaigns and customer service improvements.

AIM

To gauge public sentiment towards the brand or products for strategic insights.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

```
import os
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

CSV_PATH = "opinions.csv"
df = pd.read_csv(CSV_PATH)

assert "text" in df.columns, "CSV must include a 'text' column."

for c in ["brand", "product", "platform", "date"]:
    if c not in df.columns:
        df[c] = np.nan

if df["date"].notna().any():
```

```
    df["date"] = pd.to_datetime(df["date"], errors="coerce")
def clean_text(s: str) -> str:
    if not isinstance(s, str):
        return ""
    s = s.strip()

    s = re.sub(r"http\S+|www\.\S+", " ", s)
    s = re.sub(r"\s+", " ", s)
    return s

df["text_clean"] = df["text"].apply(clean_text)

df = df[df["text_clean"].str.len() > 0].copy()

import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

try:
    _ = nltk.data.find("sentiment/vader_lexicon.zip")
except LookupError:
    nltk.download("vader_lexicon")

sia = SentimentIntensityAnalyzer()
scores = df["text_clean"].apply(sia.polarity_scores).apply(pd.Series)

df["sentiment_score"] = scores["compound"]
def label_sentiment(c):
    if c >= 0.3:
        return "positive"
    elif c <= -0.3:
        return "negative"
    else:
        return "neutral"

df["sentiment_label"] = df["sentiment_score"].apply(label_sentiment)

overall =
    df["sentiment_label"].value_counts(normalize=True).reindex( ["positive",
    "neutral", "negative"]
).fillna(0) * 100

by_brand =
df.groupby("brand")["sentiment_score"].mean().sort_values(ascending=False)
mix_by_brand = (
    df.pivot_table(index="brand", columns="sentiment_label",
```

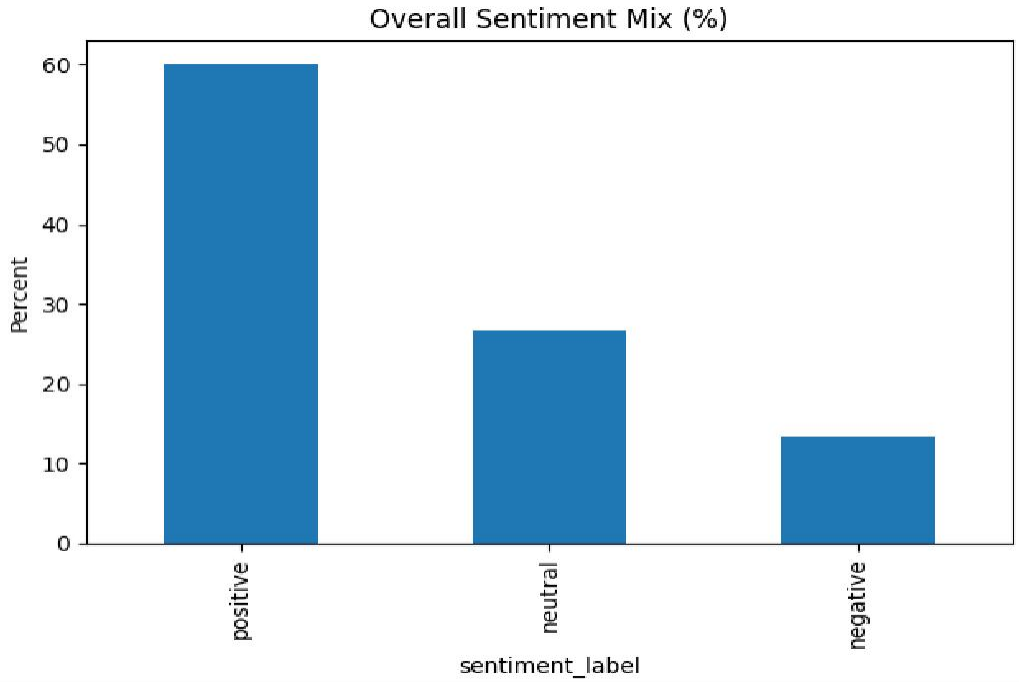
```
    values="text_clean", aggfunc="count")
```

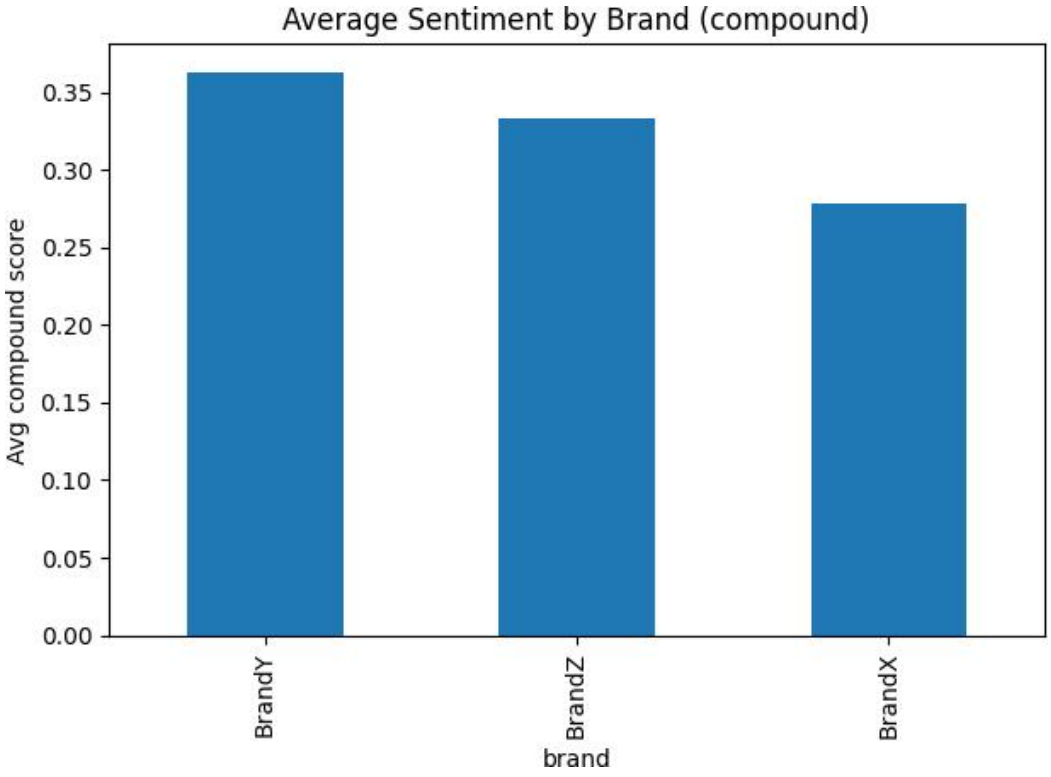
```
    .fillna(0)
    .reindex(columns=["positive", "neutral", "negative"])
    .astype(int)
)
for col in mix_by_brand.columns:
    mix_by_brand[col + "_pct"] = (mix_by_brand[col] /
mix_by_brand.sum(axis=1) * 100).round(1)

by_product =
df.groupby("product")["sentiment_score"].mean().sort_values(ascending=Fals
e)
by_platform =
df.groupby("platform")["sentiment_score"].mean().sort_values(ascending=Fal
se)

trend = None
if df["date"].notna().any():
    trend = (
        df.dropna(subset=["date"])
          .set_index("date")
          .resample("D")["sentiment_score"]
          .mean()
          .to_frame("avg_sentiment")
          .fillna(method="ffill")
    )
top_pos = df.sort_values("sentiment_score",
    ascending=False).head(10)[ ["text", "brand", "product", "platform",
    "sentiment_score"]
]
top_neg = df.sort_values("sentiment_score",
    ascending=True).head(10)[ ["text", "brand", "product", "platform",
    "sentiment_score"]
]

os.makedirs("sentiment_reports", exist_ok=True)

df.to_csv("sentiment_reports/scored_rows.csv", index=False)
overall.to_csv("sentiment_reports/overall_mix_percent.csv",
header=["percent"])
mix_by_brand.to_csv("sentiment_reports/mix_by_brand.csv")
by_brand.to_csv("sentiment_reports/avg_score_by_brand.csv",
header=["avg_sentiment"])
by_product.to_csv("sentiment_reports/avg_score_by_product.csv",
header=["avg_sentiment"])
by_platform.to_csv("sentiment_reports/avg_score_by_platform.csv",
```

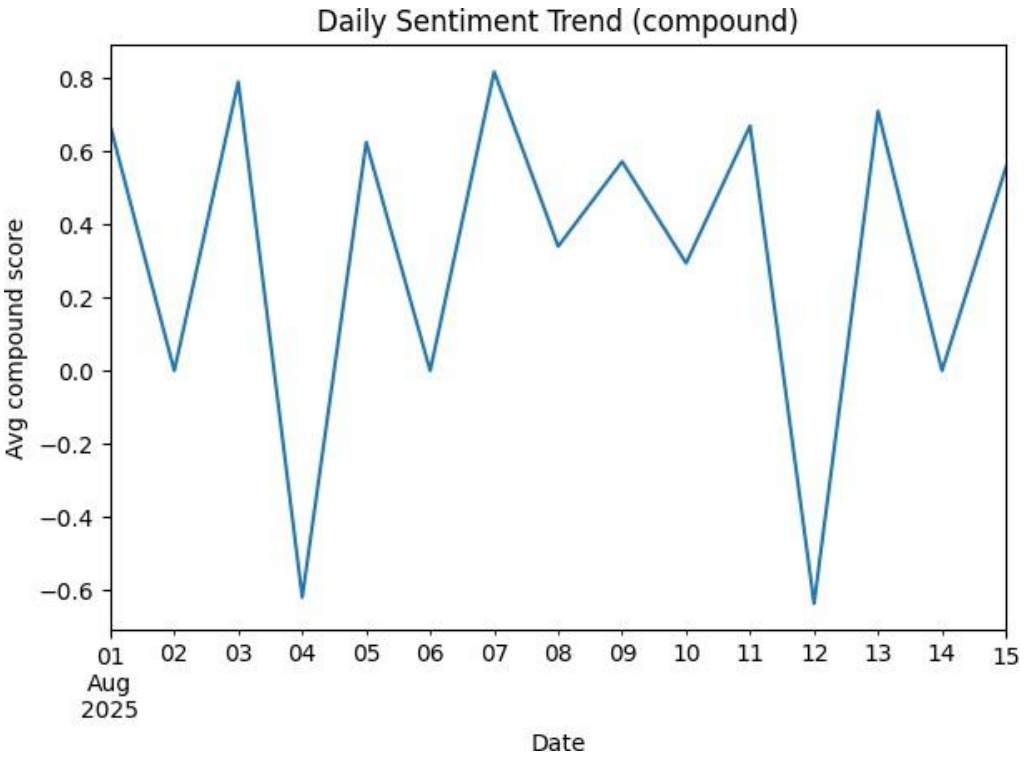
```
    header=["avg_sentiment"])
```

```
if trend is not None:
    trend.to_csv("sentiment_reports/daily_trend.csv")

plt.figure()
overall.plot(kind="bar")
plt.title("Overall Sentiment Mix (%)")
plt.ylabel("Percent")
plt.tight_layout()
plt.savefig("sentiment_reports/overall_mix.png", dpi=160)

plt.figure()
by_brand.plot(kind="bar")
plt.title("Average Sentiment by Brand (compound)")
plt.ylabel("Avg compound score") plt.tight_layout()
plt.savefig("sentiment_reports/avg_by_brand.png", dpi=160)

if trend is not None: plt.figure()
    trend["avg_sentiment"].plot()
    plt.title("Daily Sentiment Trend (compound)")
    plt.ylabel("Avg compound score")
    plt.xlabel("Date")
    plt.tight_layout()
    plt.savefig("sentiment_reports/daily_trend.png", dpi=160)

top_pos.to_csv("sentiment_reports/top_positive_examples.csv", index=False)
top_neg.to_csv("sentiment_reports/top_negative_examples.csv", index=False)

print("Done. See the 'sentiment_reports' folder for CSVs and PNG charts.")
```

OUTPUT


Overall Sentiment Mix (%)
Percent
60
50
40
30
20
10
0
positive
neutral
negative
sentiment_label


Average Sentiment by Brand (compound)
Avg compound score
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
BrandY
BrandZ
BrandX
brand


Daily Sentiment Trend (compound)
Avg compound score
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
01
Aug
2025
02
03
04
05
06
07
08
09
10
11
12
13
14
15
Date

Criteria	Marks
Preparation	**/20**
Observation	**/25**
Interpretation of result	**/20**
viva	**/10**
Total	**/75**
Faculty signature with date	

RESULT

Thus, using sentiment analysis, brand perception has been enhanced successfully.

EX.NO: 5	Fraud Detection

OBJECTIVE

To identify and prevent fraudulent transactions by analyzing patterns and anomalies in transaction data, safeguarding financial assets and customer trust.

READING MATERIAL:

Fraud detection is the process of identifying and preventing fraudulent activities, such as unauthorized transactions, identity theft, or money laundering. It is a critical aspect of financial services, e-commerce, and other industries where financial transactions are common. Fraud detection systems aim to protect businesses and customers from financial losses and maintain trust.

These systems use data analysis and machine learning algorithms to detect patterns and anomalies in transaction data that may indicate fraudulent behavior. Techniques include:

- Anomaly Detection: Identifying transactions that deviate significantly from normal patterns, which may indicate fraud.
- Supervised Learning Models: Training models on labeled data (fraudulent vs. non-fraudulent transactions) to classify new transactions.
- Rule-Based Systems: Implementing business rules that trigger alerts for specific conditions, such as multiple large transactions in a short period.

For instance, a credit card company might use fraud detection models to analyze transaction data in real-time. If the system detects an unusual purchase, such as a high-value transaction in a different country, it can flag the transaction for further investigation or temporarily block the card to prevent potential fraud.

EX.NO: 5	Fraud Detection

OBJECTIVE

To identify and prevent fraudulent transactions by analyzing patterns and anomalies in transaction data, safeguarding financial assets and customer trust.

AIM

To identify and prevent fraudulent activities in transactions.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

```
import os
import math
import joblib
import numpy as np
import pandas as pd
from datetime import datetime
from typing import List

from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_auc_score,
    precision_recall_curve, average_precision_score
)
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import IsolationForest
from sklearn.utils.class_weight import compute_class_weight
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline

CSV = "transactions.csv"
df = pd.read_csv(CSV)

if "timestamp" in df.columns:
    df["timestamp"] = pd.to_datetime(df["timestamp"], errors="coerce")

if "timestamp" in df.columns:
    df["hour"]       = df["timestamp"].dt.hour  df["dayofweek"]
    = df["timestamp"].dt.dayofweek df["month"]   =
    df["timestamp"].dt.month df["is_weekend"] =
    df["dayofweek"].isin([5,6]).astype(int)
else:
    for c in ["hour","dayofweek","month","is_weekend"]:
        if c not in df.columns:
            df[c] = np.nan


if "customer_id" in df.columns and "timestamp" in df.columns:
    df = df.sort_values(["customer_id", "timestamp"])
    df["prev_ts"] = df.groupby("customer_id")["timestamp"].shift(1)
```

```
    df["secs_since_prev_txn"] = (df["timestamp"] -
df["prev_ts"]).dt.total_seconds()
    df["secs_since_prev_txn"] =
df["secs_since_prev_txn"].fillna(df["secs_since_prev_txn"].median())
else:
    df["secs_since_prev_txn"] = df["secs_since_prev_txn"] if
"secs_since_prev_txn" in df.columns else 0.0

if "amount" in df.columns:
    df["amount_log1p"] = np.log1p(df["amount"].clip(lower=0))
else:
    raise ValueError("CSV must include an 'amount' column.")

def freq_encode(series: pd.Series) -> pd.Series:
    freq = series.value_counts(dropna=False)
    return series.map(freq).astype(float)

cat_cols: List[str] = [c for c in
["merchant_id","customer_id","category","device","ip"] if c in df.columns]
for c in cat_cols:
    df[f"{c}_freq"] = freq_encode(df[c].astype(str))

num_features = [
    "amount_log1p", "hour", "dayofweek", "month", "is_weekend",
    "secs_since_prev_txn"
] + [f"{c}_freq" for c in cat_cols]

X = df[num_features].fillna(0.0)

supervised = "label" in df.columns and
set(df["label"].dropna().unique()).issubset({0,1})

os.makedirs("fraud_outputs", exist_ok=True)

if supervised:
    y = df["label"].astype(int).values

    X_train, X_test, y_train, y_test, df_train_idx, df_test_idx =
train_test_split(
        X, y, df.index, test_size=0.25, random_state=42, stratify=y
    )
    classes = np.array([0,1])
    weights = compute_class_weight(class_weight="balanced",
classes=classes, y=y_train)
    class_weight = {0: float(weights[0]), 1: float(weights[1])}
```

```
    pipe =
        ImbPipeline(steps=[ ("scaler",
        StandardScaler()),
        ("smote", SMOTE(random_state=42, k_neighbors=5)),
        ("clf", LogisticRegression(
            solver="liblinear",
            class_weight=class_weight,
            max_iter=200,
            random_state=42
        ))
    ])

    pipe.fit(X_train, y_train)


    proba = pipe.predict_proba(X_test)[:,1]
    precision, recall, thresholds = precision_recall_curve(y_test, proba)
    f1 = (2*precision*recall) / (precision+recall + 1e-12)
    best_idx = int(np.nanargmax(f1))
    best_threshold = 0.5 if best_idx >= len(thresholds) else
thresholds[best_idx]

    y_pred = (proba >= best_threshold).astype(int)


    ap = average_precision_score(y_test, proba)
    roc = roc_auc_score(y_test, proba)
    cm = confusion_matrix(y_test, y_pred)

    print("=== Supervised Logistic Regression (with SMOTE + class_weight)
===")
    print(f"Best threshold (F1-optimal): {best_threshold:0.4f}")
    print("Confusion matrix [ [TN FP] [FN TP] ]:")
    print(cm)
    print("\nClassification report:")
    print(classification_report(y_test, y_pred, digits=4))
    print(f"ROC-AUC: {roc:0.4f}")
    print(f"Average Precision (PR-AUC): {ap:0.4f}")


    joblib.dump({"model": pipe, "features": num_features, "threshold":
best_threshold}, "fraud_outputs/supervised_lr.joblib")
```

```
        out = df.loc[df_test_idx, :].copy()
```

```
    out["fraud_proba"] = proba
    out["fraud_pred"]  = y_pred
    out.to_csv("fraud_outputs/scored_test_supervised.csv", index=False)
    print("Saved: fraud_outputs/supervised_lr.joblib,
fraud_outputs/scored_test_supervised.csv")
else:
    pipe = Pipeline(steps=[ ("scaler",
        StandardScaler()), ("iforest",
        IsolationForest(
            n_estimators=300,
            contamination="auto",
            random_state=42,
            n_jobs=-1
        ))
    ])

    pipe.fit(X)
    pred_iso = pipe["iforest"].predict(pipe["scaler"].transform(X))
    score_iso =
pipe["iforest"].decision_function(pipe["scaler"].transform(X))
    s = (score_iso - score_iso.min()) / (score_iso.max() - score_iso.min()
+ 1e-12)
    fraud_score = 1.0 - s
    top_pct = 0.01
    thr = np.quantile(fraud_score, 1.0 - top_pct)
    fraud_flag = (fraud_score >= thr).astype(int)

    print("=== Unsupervised Isolation Forest ===")
    print(f"Flagging approximately top {int(top_pct*100)}% most suspicious
as fraud.")
    print(f"Threshold on fraud_score: {thr:0.4f}")
    print(f"Total flagged: {fraud_flag.sum()} / {len(fraud_flag)}")

    joblib.dump({"model": pipe, "features": num_features,
"heuristic_threshold": float(thr)},
                "fraud_outputs/unsupervised_iforest.joblib")
    out = df.copy()
    out["fraud_score_0to1"] = fraud_score
    out["fraud_flag"] = fraud_flag
    out.to_csv("fraud_outputs/scored_unsupervised.csv", index=False)
    print("Saved: fraud_outputs/unsupervised_iforest.joblib,
fraud_outputs/scored_unsupervised.csv")

print("\nDone. Check the 'fraud_outputs' folder for models and scored
CSVs.")
```

OUTPUT

```
=== Supervised Logistic Regression (with SMOTE + class_weight) ===
Best threshold (F1-optimal): 0.9907
Confusion matrix [ [TN FP] [FN TP] ]:
[[43  3]
 [ 3  1]]

Classification report:
              precision    recall  f1-score   support

           0     0.9348    0.9348    0.9348        46
           1     0.2500    0.2500    0.2500         4

    accuracy                         0.8800        50
   macro avg     0.5924    0.5924    0.5924        50
weighted avg     0.8800    0.8800    0.8800        50

ROC-AUC: 0.4130
Average Precision (PR-AUC): 0.1207
Saved: fraud_outputs/supervised_lr.joblib, fraud_outputs/scored_test_supervised.csv

Done. Check the 'fraud_outputs' folder for models and scored CSVs.
```

Criteria	Marks
Preparation	**/20**
Observation	**/25**
Interpretation of result	**/20**
viva	**/10**
Total	**/75**
Faculty signature with date	

RESULT

Thus, using fraud detection systems, fraudulent transactions have been minimized successfully.

EX.NO: 6	A/B Testing for Website Optimization

OBJECTIVE

To evaluate the effectiveness of different website designs or features by comparing user interactions, leading to data-driven decisions for enhancing user experience and conversion rates.

READING MATERIAL:

A/B testing, or split testing, is a method used to compare two versions of a webpage or user experience to determine which one performs better in terms of key metrics like conversion rates, click-through rates, or user engagement. In A/B testing, users are randomly assigned to two groups: Group A sees the original version (control), while Group B sees the modified version (variant).

The goal is to identify which version leads to better user outcomes. A/B testing is commonly used in digital marketing, product development, and user experience design to optimize website elements such as headlines, images, call-to-action buttons, layouts, or color schemes.

For example, an e-commerce website might conduct an A/B test to determine if a new checkout page design leads to a higher conversion rate. The company can analyze the results to see if the variant outperforms the control in terms of completed purchases, and then implement the winning version to improve overall sales.

EX.NO: 6	A/B Testing for Website Optimization

OBJECTIVE

To evaluate the effectiveness of different website designs or features by comparing user interactions, leading to data-driven decisions for enhancing user experience and conversion rates.

AIM

To determine the best website design for improving user engagement and conversions.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

OUPUT:

RESULT

Thus, using A/B testing, website conversion rates have been increased successfully.

EX.NO: 7	Employee Performance Analysis

OBJECTIVE

To assess the productivity and effectiveness of employees using key performance indicators (KPIs), supporting HR decisions related to promotions, training, and resource allocation.

READING MATERIAL:

Employee performance analysis involves assessing the productivity, efficiency, and effectiveness of employees within an organization. This analysis helps identify top performers, areas where employees may need improvement, and overall workforce trends. It is used to inform decisions related to promotions, training, resource allocation, and performance management.

Key performance indicators (KPIs) are often used to measure employee performance. These can include:

- Quantitative Metrics: Such as sales numbers, customer service scores, project completion rates, or production output.
- Qualitative Metrics: Such as peer reviews, supervisor evaluations, or customer feedback.

For example, a sales department might analyze metrics like total sales, average deal size, and customer acquisition rates to assess the performance of individual sales representatives. This analysis can help identify high-performing reps for recognition or promotion and determine training needs for others.

EX.NO: 7	Employee Performance Analysis

OBJECTIVE

To assess the productivity and effectiveness of employees using key performance indicators (KPIs), supporting HR decisions related to promotions, training, and resource allocation.

AIM

To assess and improve employee productivity and effectiveness.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

OUTPUT

RESULT

Thus, using employee performance analysis, productivity improvements have been implemented successfully

EX.NO: 8	Supply Chain Optimization

OBJECTIVE

To streamline supply chain processes by analyzing logistics, inventory, and production data, reducing costs and improving efficiency.

READING MATERIAL:

Supply chain optimization involves improving the efficiency and effectiveness of a company's supply chain operations. This includes optimizing processes related to procurement, production, transportation, warehousing, and distribution. The goal is to minimize costs, reduce lead times, and ensure timely delivery of products to customers.

Key aspects of supply chain optimization include:

- Demand Forecasting: Predicting customer demand to plan inventory levels and production schedules accurately.
- Inventory Management: Balancing inventory levels to avoid overstocking or stockouts, optimizing storage costs, and ensuring product availability.
- Transportation Planning: Choosing the most cost-effective and efficient shipping methods and routes.

For instance, a manufacturer might use supply chain optimization techniques to streamline its production process, reduce raw material costs, and improve delivery times. By doing so, the company can lower operational costs and enhance customer satisfaction.

EX.NO: 8	Supply Chain Optimization

OBJECTIVE

To streamline supply chain processes by analyzing logistics, inventory, and production data, reducing costs and improving efficiency.

AIM

To enhance efficiency and reduce costs in supply chain operations.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

OUTPUT

RESULT

Thus, using supply chain optimization strategies, operational costs and delays have been reduced successfully.

EX.NO: 9	Customer Lifetime Value Prediction

OBJECTIVE

To estimate the total revenue a customer will generate over their relationship with the company, enabling more informed decisions on customer acquisition and retention strategies.

READING MATERIAL:

Customer Lifetime Value (CLV) prediction estimates the total revenue a customer is expected to generate for a company over the entire duration of their relationship. CLV is a crucial metric for businesses as it helps determine the value of acquiring and retaining customers, guiding marketing and customer service investments.

CLV is calculated by analyzing factors such as:

- Average Purchase Value: The average amount spent by a customer per transaction.
- Purchase Frequency: How often the customer makes a purchase.
- Customer Lifespan: The expected duration of the customer's relationship with the company.

For example, a subscription-based service might calculate CLV by considering the average monthly subscription fee, the average number of months a customer stays subscribed, and any additional purchases made. This information helps the company decide how much to invest in customer acquisition and retention efforts.

EX.NO: 9	Customer Lifetime Value Prediction

OBJECTIVE

To estimate the total revenue a customer will generate over their relationship with the company, enabling more informed decisions on customer acquisition and retention strategies.

AIM

To estimate the total value a customer brings over their relationship with the company.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

OUTPUT

RESULT

Thus, using customer lifetime value prediction, customer retention has been increased successfully.

EX.NO: 10	Market Basket Analysis

OBJECTIVE

To identify associations between products purchased together, informing cross-selling and upselling strategies to increase average transaction value.

READING MATERIAL:

Market basket analysis is a data mining technique used to identify associations between products that are frequently bought together. It helps businesses understand customer purchasing patterns and design effective cross-selling and upselling strategies. The analysis uses techniques such as association rule mining and frequent itemset mining to discover relationships between items.

Key concepts in market basket analysis include:

- **Support**: The frequency with which an itemset appears in transactions.
- **Confidence**: The likelihood that a customer buys an item, given that they have already bought another item.
- **Lift**: The strength of the association between items, indicating how much more likely they are to be bought together than individually.

For instance, a grocery store might use market basket analysis to identify that customers who buy bread are also likely to buy butter. The store can then place these items near each other or offer discounts on complementary products to increase sales.

EX.NO: 10	Market Basket Analysis

OBJECTIVE

To identify associations between products purchased together, informing cross-selling and upselling strategies to increase average transaction value.

AIM

To identify product purchase patterns for better cross-selling and upselling strategies.

ALGORITHM / PROCEDURE

Step 1: Importing Libraries

Step 2: Exploration of Data

Step 3: Data Visualization

Step 4: Data Preprocessing

Step 5: Splitting Data

Step 6: Model Selection and Training

Step 7: Model Evaluation

PROGRAM

OUTPUT

RESULT

Thus, using market basket analysis, cross-selling and upselling strategies have been enhanced successfully.