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

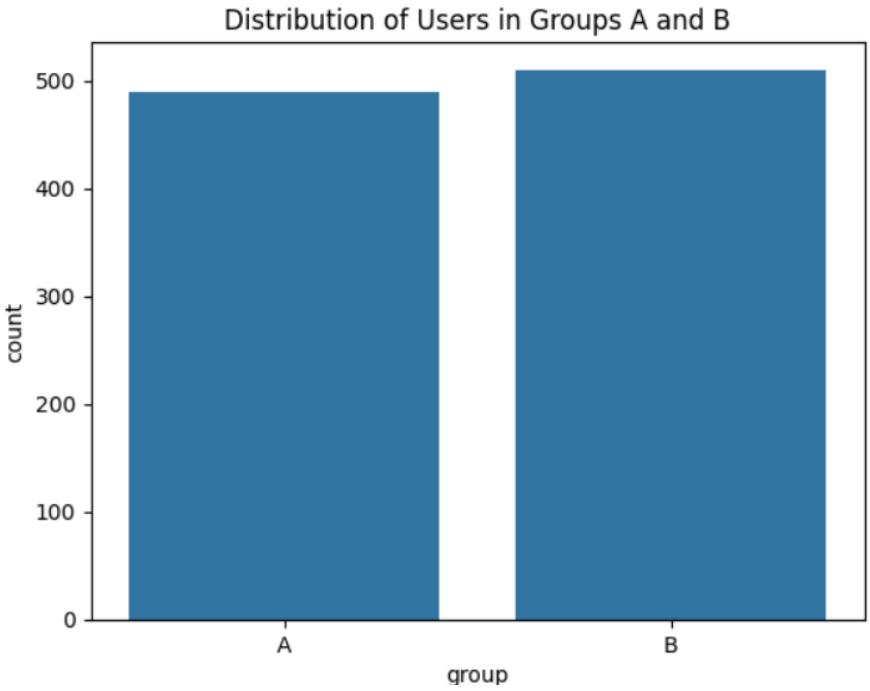
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
np.random.seed(42)
n = 1000
# Randomly assign users to group A or B
groups = np.random.choice(['A', 'B'], size=n)

# Simulate conversion probabilities
conversion_probs = np.where(groups == 'A', 0.10, 0.15)  # B performs slightly better
# Generate whether user converted or not
```

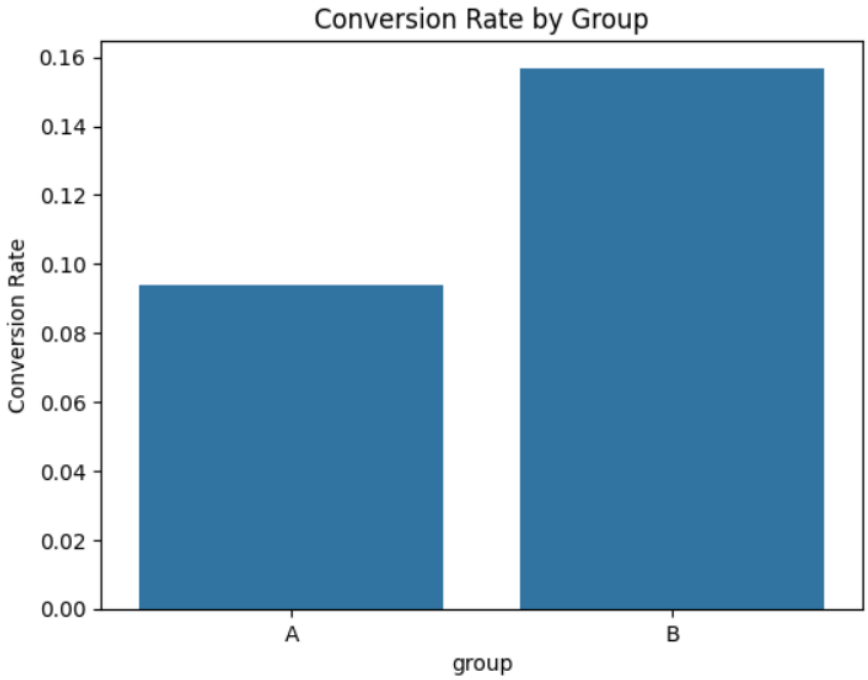
```
converted = np.random.binomial(1, conversion_probs)
# Additional features (optional)
time_on_page = np.random.normal(45, 10, n) + (converted * 5)
click_rate = np.random.normal(0.05, 0.02, n) + (converted * 0.03)
# Create DataFrame
data = pd.DataFrame({
    'user_id': np.arange(1, n+1),
    'group': groups,
    'time_on_page': time_on_page,
    'click_rate': click_rate,
    'converted': converted
})
print("First 5 rows of the dataset:")
print(data.head(), "\n")
print("Dataset info:")
print(data.info()
# Count of users in each group
sns.countplot(x='group', data=data)
plt.title('Distribution of Users in Groups A and B')
plt.show()
# Conversion rate by group
conversion_rates = data.groupby('group')['converted'].mean().reset_index()
sns.barplot(x='group', y='converted', data=conversion_rates)
plt.title('Conversion Rate by Group')
plt.ylabel('Conversion Rate')
plt.show()
# Time on page vs conversion
sns.boxplot(x='group', y='time_on_page', data=data)
plt.title('Time on Page by Group')
plt.show()
# Encode group: A=0, B=1
data['group'] = data['group'].map({'A': 0, 'B': 1})
# Drop any missing values (if exist)
data.dropna(inplace=True)
X = data[['group', 'time_on_page', 'click_rate']]  # independent variables
```

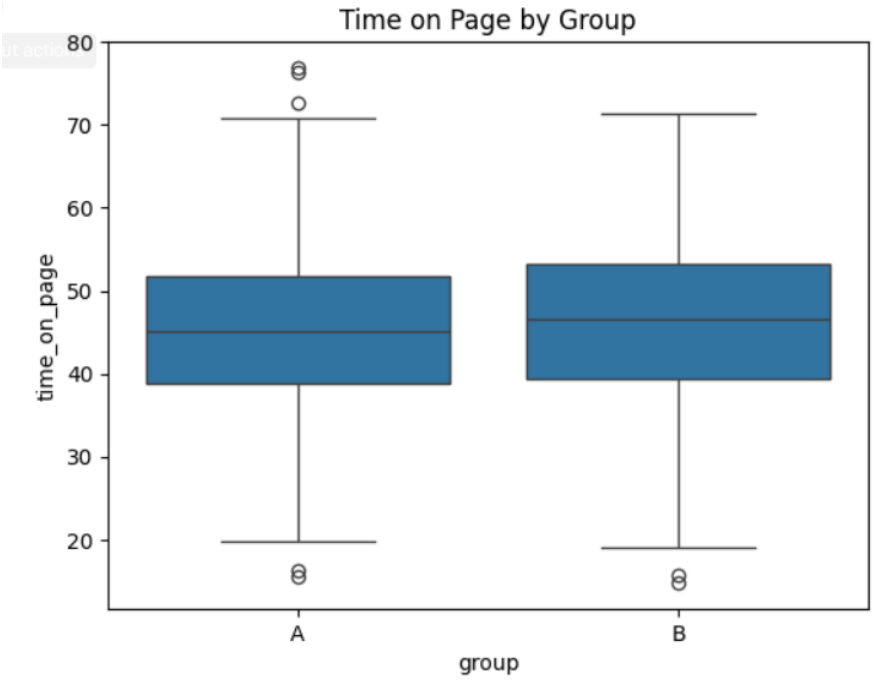
```
y = data['converted']  # target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("\n--- Model Evaluation ---")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# Check model coefficients
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_[0]
})
print("\nFeature Importance:")
print(coef_df)
if coef_df.loc[coef_df['Feature'] == 'group', 'Coefficient'].values[0] > 0:
    print("\n Version B leads to a higher likelihood of conversion.")
else:
    print("\n Version A performs better for conversions.")
# Compare actual vs predicted conversions
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
sns.heatmap(pd.crosstab(comparison['Actual'], comparison['Predicted']), annot=True, fmt='d',
cmap='Blues')
plt.title('Actual vs Predicted Conversions')
plt.show()
```

OUTPUT:

```
First 5 rows of the dataset:
   user_id group  time_on_page  click_rate  converted
0        1     A     20.005943    0.027771          0
1        2     B     67.909426    0.054930          0
2        3     A     31.104275    0.059964          0
3        4     A     28.546013    0.072803          0
4        5     A     55.225704    0.081611          0

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   user_id       1000 non-null   int64
 1   group         1000 non-null   object
 2   time_on_page  1000 non-null   float64
 3   click_rate    1000 non-null   float64
 4   converted     1000 non-null   int64
dtypes: float64(2), int64(2), object(1)
memory usage: 39.2+ KB
None
```




Conversion Rate by Group
Conversion Rate
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
A
B
group


Time on Page by Group
time_on_page
80
70
60
50
40
30
20
A
B
group

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

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

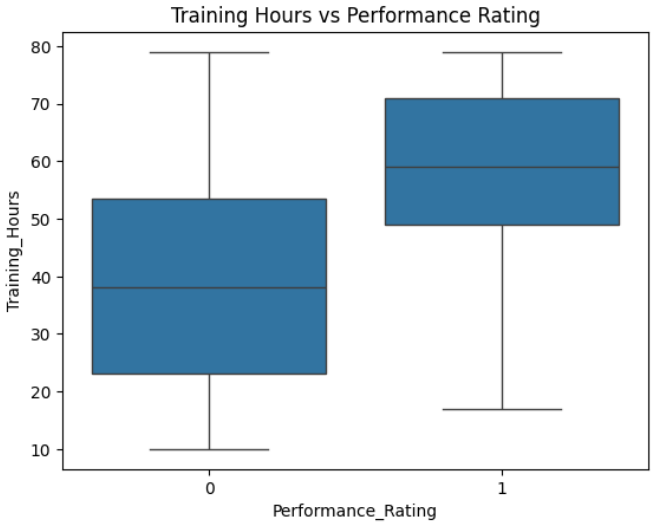
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
np.random.seed(42)
n = 500
employee_id = np.arange(1, n+1)
age = np.random.randint(22, 60, n)
department = np.random.choice(['HR', 'Sales', 'IT', 'Finance', 'Operations'], n)
experience_years = np.random.randint(1, 20, n)
training_hours = np.random.randint(10, 80, n)
projects_completed = np.random.randint(1, 10, n)
```

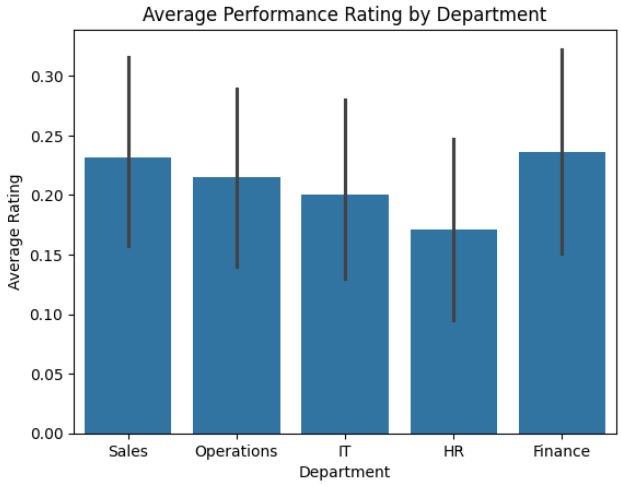
```
avg_monthly_hours = np.random.randint(120, 250, n)
absences = np.random.randint(0, 10, n)
satisfaction_score = np.random.uniform(0.3, 1.0, n)
# Simulate performance rating (target variable)
performance_rating = (
    0.3 * (experience_years / 20)
    + 0.25 * (training_hours / 80)
    + 0.2 * (projects_completed / 10)
    + 0.15 * satisfaction_score
    - 0.1 * (absences / 10)
)
performance_rating = np.where(performance_rating > 0.55, 1, 0)   # 1=High Performer, 0=Low
Performer
# Create DataFrame
data = pd.DataFrame({
    'Employee_ID': employee_id,
    'Age': age,
    'Department': department,
    'Experience_Years': experience_years,
    'Training_Hours': training_hours,
    'Projects_Completed': projects_completed,
    'Avg_Monthly_Hours': avg_monthly_hours,
    'Absences': absences,
    'Satisfaction_Score': satisfaction_score,
    'Performance_Rating': performance_rating
})
print("First 5 rows of dataset:")
print(data.head(), "\n")
print("Dataset info:")
print(data.info())
sns.countplot(x='Performance_Rating', data=data)
plt.title("Distribution of Employee Performance Ratings")
plt.show()
# Department vs Performance
sns.barplot(x='Department', y='Performance_Rating', data=data, estimator=np.mean)
plt.title("Average Performance Rating by Department")
plt.ylabel("Average Rating")
plt.show()
# Satisfaction vs Performance
sns.boxplot(x='Performance_Rating', y='Satisfaction_Score', data=data)
plt.title("Satisfaction vs Performance Rating")
plt.show()
# Training Hours vs Performance
sns.boxplot(x='Performance_Rating', y='Training_Hours', data=data)
```

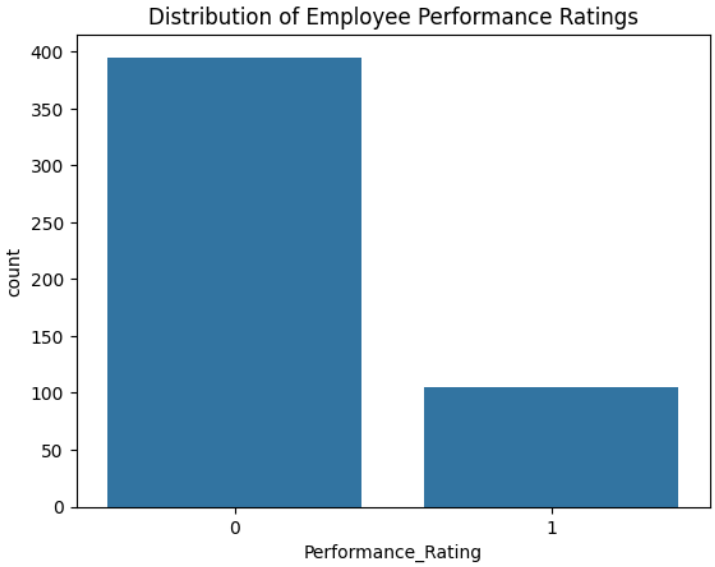
```
plt.title("Training Hours vs Performance Rating")
plt.show()
data = pd.get_dummies(data, columns=['Department'], drop_first=True)
# Define features and target
X = data.drop(['Employee_ID', 'Performance_Rating'], axis=1)
y = data['Performance_Rating']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("\n--- Model Evaluation ---")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# Feature importance
importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:")
print(importances)
plt.figure(figsize=(8, 5))
sns.barplot(x='Importance', y='Feature', data=importances.head(10))
plt.title("Top 10 Features Influencing Employee Performance")
plt.show()
top_feature = importances.iloc[0]['Feature']
print(f"\n Insight: The most influential factor in predicting performance is **{top_feature}**.")
```

OUTPUT

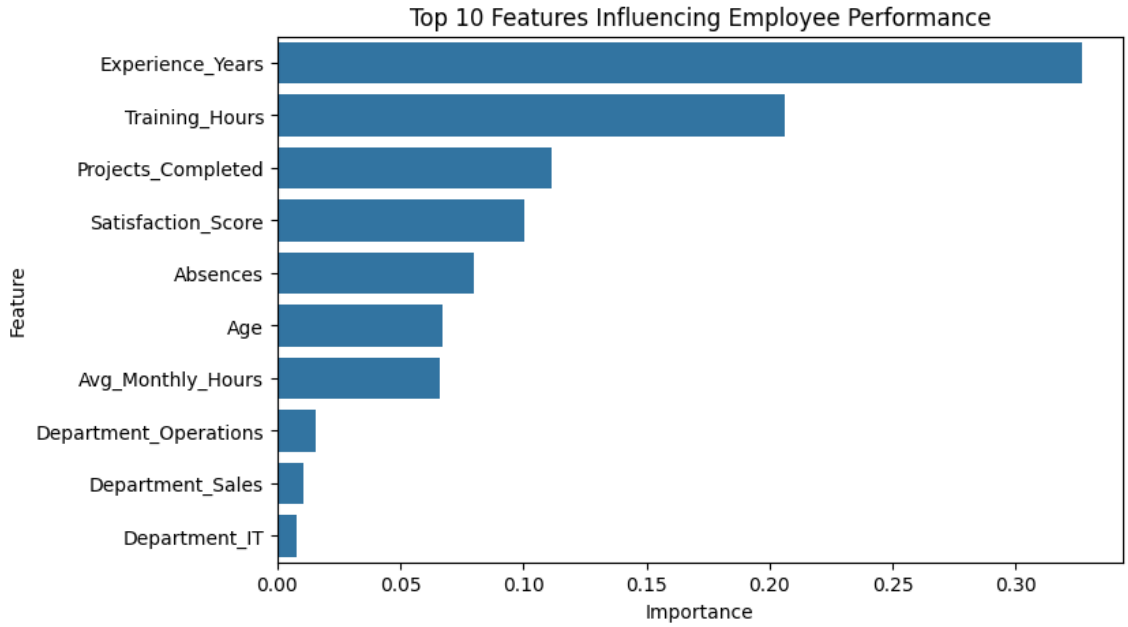
```
First 5 rows of dataset:
   Employee_ID  Age  Department  Experience_Years  Training_Hours  \
0            1   50       Sales                 6              73
1            2   36       Sales                 1              30
2            3   29  Operations                17              27
3            4   42  Operations                11              73
4            5   40  Operations                18              47

   Projects_Completed  Avg_Monthly_Hours  Absences  Satisfaction_Score  \
0                   2                235         0            0.749214
1                   7                179         3            0.770035
2                   3                240         8            0.742598
3                   3                227         8            0.439294
4                   5                227         4            0.592834

   Performance_Rating
0                   0
1                   0
2                   0
3                   0
4                   1
```







```
--- Model Evaluation ---
Accuracy: 0.92

Confusion Matrix:
 [[81  0]
 [ 8 11]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        81
           1       1.00      0.58      0.73        19

    accuracy                           0.92       100
   macro avg       0.96      0.79      0.84       100
weighted avg       0.93      0.92      0.91       100


Feature Importance:
                  Feature  Importance
1        Experience_Years    0.327217
2          Training_Hours    0.206444
3      Projects_Completed    0.111538
6      Satisfaction_Score    0.100097
5                Absences    0.079911
0                     Age    0.067026
4       Avg_Monthly_Hours    0.065939
9   Department_Operations    0.015370
10       Department_Sales    0.010590
8           Department_IT    0.008029
7           Department_HR    0.007840
```



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

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

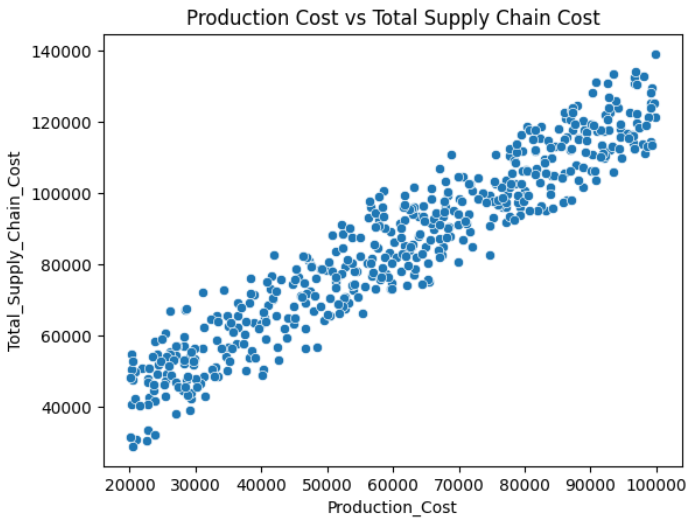
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
np.random.seed(42)
n = 500
data = pd.DataFrame({
    'Factory_ID': np.random.randint(1, 10, n),
    'Region': np.random.choice(['North', 'South', 'East', 'West'], n),
    'Production_Cost': np.random.uniform(20000, 100000, n),
    'Transportation_Cost': np.random.uniform(5000, 30000, n),
    'Inventory_Cost': np.random.uniform(1000, 10000, n),
```

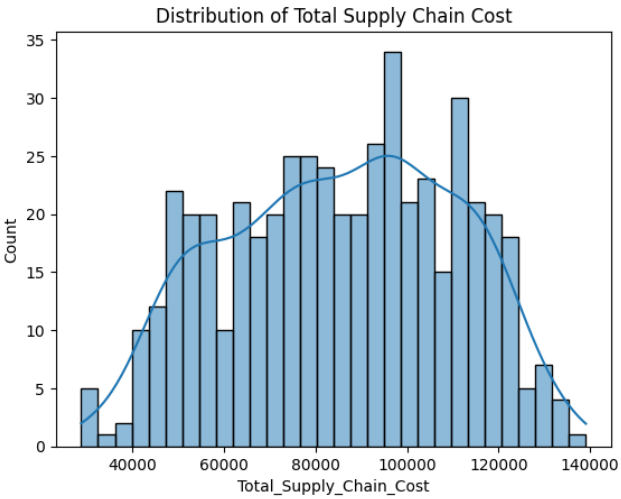
```
    'Demand': np.random.uniform(1000, 10000, n),
    'Lead_Time': np.random.uniform(1, 10, n),
    'On_Time_Delivery_Rate': np.random.uniform(0.6, 1.0, n),
    'Supplier_Rating': np.random.uniform(2, 5, n)
})
# Target variable: Total Supply Chain Cost
data['Total_Supply_Chain_Cost'] = (
    data['Production_Cost'] + data['Transportation_Cost'] + data['Inventory_Cost']
) * (1 + (1 - data['On_Time_Delivery_Rate']) * 0.1)
print("First 5 rows of dataset:")
print(data.head(), "\n")
print("Dataset info:")
print(data.info())
sns.histplot(data['Total_Supply_Chain_Cost'], bins=30, kde=True)
plt.title("Distribution of Total Supply Chain Cost")
plt.show()
# Relationship between production and total cost
sns.scatterplot(x='Production_Cost', y='Total_Supply_Chain_Cost', data=data)
plt.title("Production Cost vs Total Supply Chain Cost")
plt.show()
# Relationship between lead time and total cost
sns.scatterplot(x='Lead_Time', y='Total_Supply_Chain_Cost', data=data)
plt.title("Lead Time vs Total Supply Chain Cost")
plt.show()
# Region-wise average total cost
sns.barplot(x='Region', y='Total_Supply_Chain_Cost', data=data, estimator=np.mean)
plt.title("Average Supply Chain Cost by Region")
plt.show()
data = pd.get_dummies(data, columns=['Region'], drop_first=True)
# Define features and target
X = data.drop(['Total_Supply_Chain_Cost'], axis=1)
y = data['Total_Supply_Chain_Cost']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("\n--- Model Evaluation ---")
print("Mean Absolute Error (MAE):", round(mae, 2))
print("Mean Squared Error (MSE):", round(mse, 2))
print("R-squared (R2):", round(r2, 3))
```

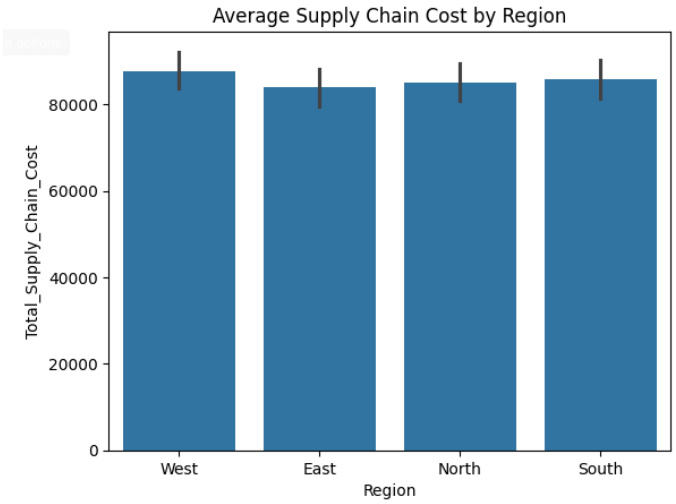
```
# Feature Importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:")
print(feature_importance)
plt.figure(figsize=(8, 5))
sns.barplot(x='Importance', y='Feature', data=feature_importance.head(10))
plt.title("Top 10 Features Affecting Supply Chain Cost")
plt.show()
top_feature = feature_importance.iloc[0]['Feature']
print(f"\n Insight: The most influential factor on total supply chain cost is **{top_feature}**.")
```

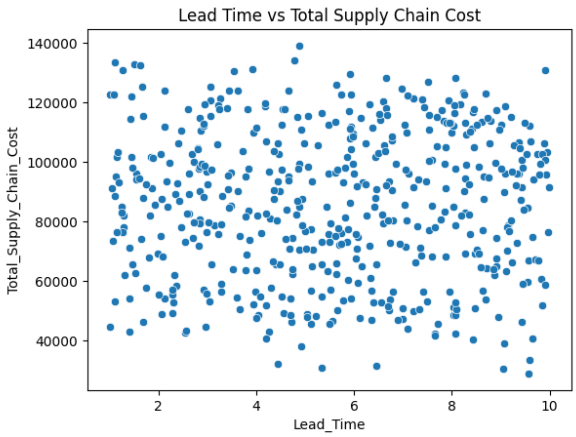
OUTPUT

```
First 5 rows of dataset:
   Factory_ID Region  Production_Cost  Transportation_Cost  Inventory_Cost  \
0           7   West     94561.147381         21189.341085     1719.688549
1           4   East     45651.375373         19859.896481     7990.512674
2           8   East     67510.647805         14590.306510     2098.315469
3           5  North     49538.438521          6640.653614     6348.616755
4           7   West     56341.437778         25537.823685     9428.042565

         Demand  Lead_Time  On_Time_Delivery_Rate  Supplier_Rating  \
0  2159.638408   3.405274               0.961423         3.911284
1  4714.850163   2.978248               0.631056         3.791877
2  8221.188962   2.183664               0.931549         3.960150
3  5311.630568   8.714166               0.739556         2.576930
4  1162.000801   6.178199               0.808640         2.886964

   Total_Supply_Chain_Cost
0            117923.337284
1             76213.586485
2             84775.626327
3             64156.205364
4             93054.562803
```

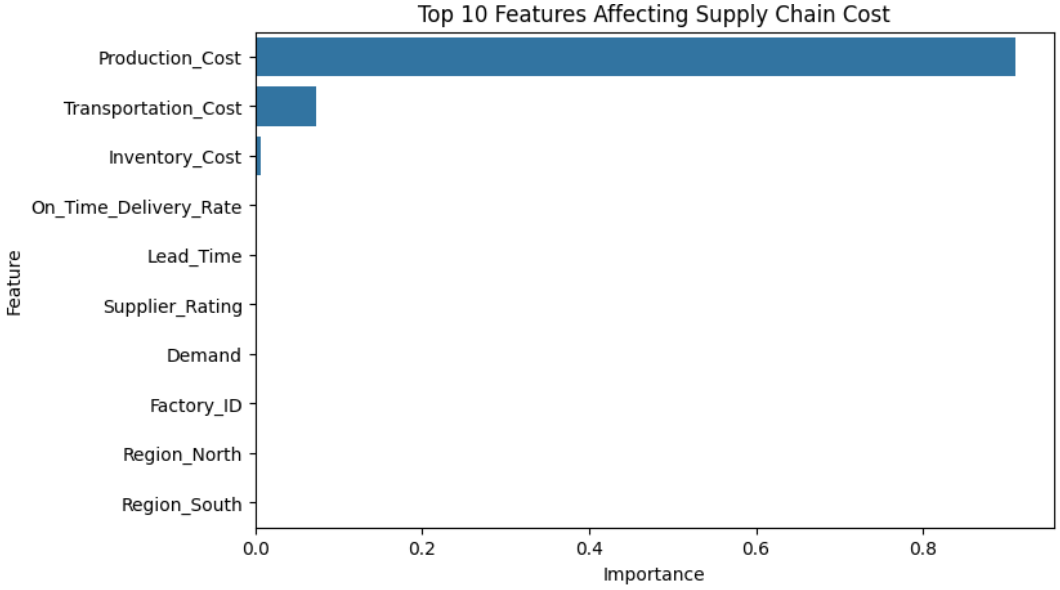








```
--- Model Evaluation ---
Mean Absolute Error (MAE): 2469.8
Mean Squared Error (MSE): 9768783.44
R-squared (R2): 0.985

Feature Importance:
                  Feature  Importance
1         Production_Cost    0.911087
2     Transportation_Cost    0.073483
3          Inventory_Cost    0.006023
6   On_Time_Delivery_Rate    0.002383
5               Lead_Time    0.002026
7         Supplier_Rating    0.001843
4                  Demand    0.001537
0              Factory_ID    0.000895
8            Region_North    0.000288
9            Region_South    0.000235
10            Region_West    0.000199
```



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

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

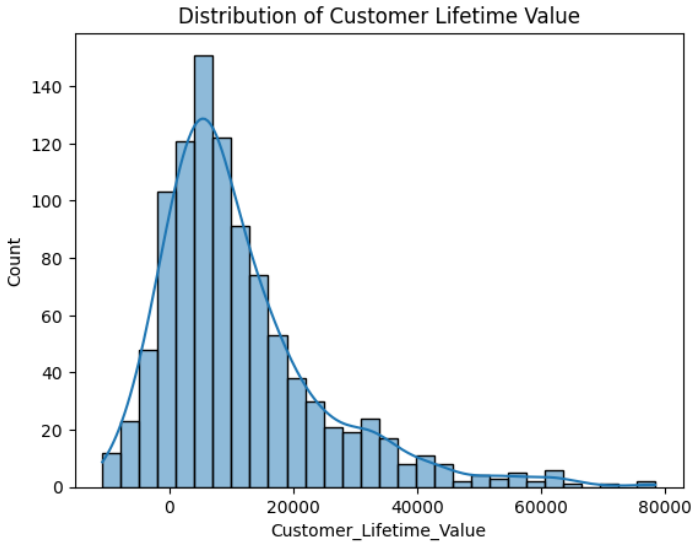
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
np.random.seed(42)
n = 1000
```

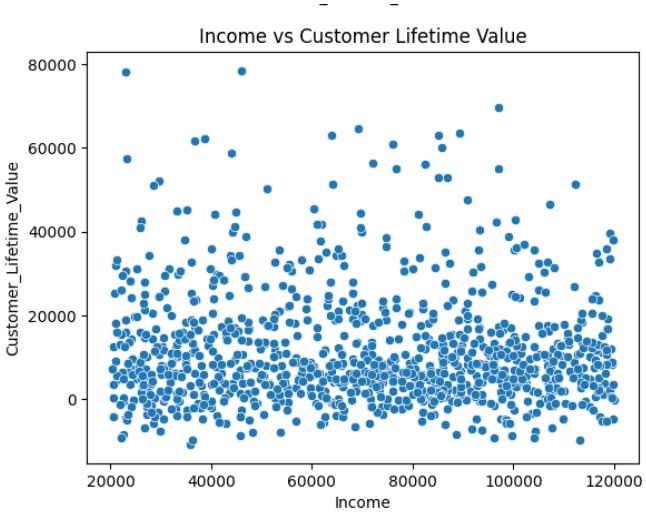
```
customer_data = pd.DataFrame({
    'Customer_ID': np.arange(1, n+1),
    'Age': np.random.randint(18, 70, n),
    'Gender': np.random.choice(['Male', 'Female'], n),
    'Income': np.random.uniform(20000, 120000, n),
    'Years_as_Customer': np.random.randint(1, 10, n),
    'Purchase_Frequency': np.random.uniform(1, 12, n),
    'Avg_Purchase_Value': np.random.uniform(50, 1000, n),
    'Customer_Satisfaction': np.random.uniform(0.4, 1.0, n)
})
# Simulate Customer Lifetime Value (Target Variable)
customer_data['Customer_Lifetime_Value'] = (
    customer_data['Purchase_Frequency'] *
    customer_data['Avg_Purchase_Value'] *
    (customer_data['Years_as_Customer'] * customer_data['Customer_Satisfaction'])
)
# Add some noise
customer_data['Customer_Lifetime_Value'] += np.random.normal(0, 5000, n)
print("First 5 rows of dataset:")
print(customer_data.head(), "\n")
print("Dataset info:")
print(customer_data.info())
# Distribution of CLV
sns.histplot(customer_data['Customer_Lifetime_Value'], bins=30, kde=True)
plt.title("Distribution of Customer Lifetime Value")
plt.show()
# CLV vs Income
sns.scatterplot(x='Income', y='Customer_Lifetime_Value', data=customer_data)
plt.title("Income vs Customer Lifetime Value")
plt.show()
# CLV vs Years as Customer
sns.scatterplot(x='Years_as_Customer', y='Customer_Lifetime_Value', data=customer_data)
plt.title("Years as Customer vs Lifetime Value")
```

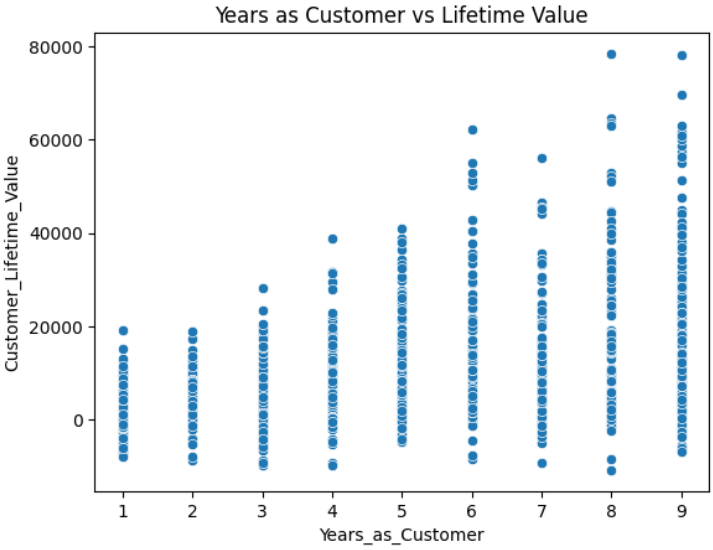
```
plt.show()
# Gender comparison
sns.barplot(x='Gender', y='Customer_Lifetime_Value', data=customer_data, estimator=np.mean)
plt.title("Average CLV by Gender")
plt.show()
customer_data = pd.get_dummies(customer_data, columns=['Gender'], drop_first=True)
# Define features and target
X = customer_data.drop(['Customer_ID', 'Customer_Lifetime_Value'], axis=1)
y = customer_data['Customer_Lifetime_Value']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("\n--- Model Evaluation ---")
print("Mean Absolute Error (MAE):", round(mae, 2))
print("Mean Squared Error (MSE):", round(mse, 2))
print("R-squared (R2):", round(r2, 3))
# Feature Importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:")
print(feature_importance)
plt.figure(figsize=(8, 5))
sns.barplot(x='Importance', y='Feature', data=feature_importance.head(10))
plt.title("Top Features Influencing Customer Lifetime Value")
plt.show()
top_feature = feature_importance.iloc[0]['Feature']
```

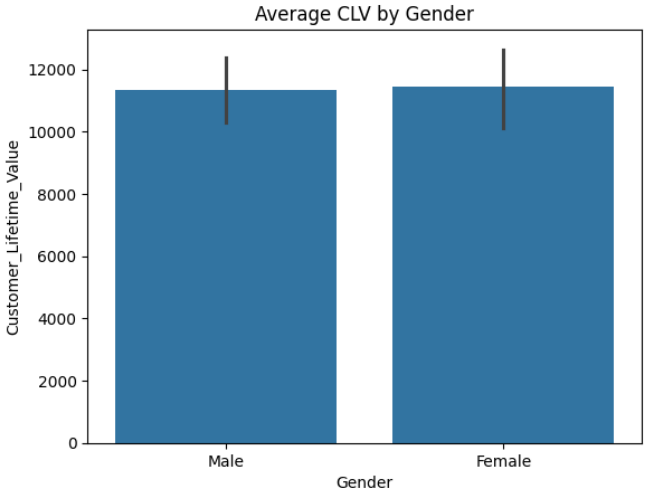
OUTPUT

```
First 5 rows of dataset:
   Customer_ID  Age  Gender         Income  Years_as_Customer  \
0            1   56    Male  105120.691405                  5
1            2   69    Male   69514.652701                  3
2            3   46    Male   68058.657733                  9
3            4   32  Female   79240.778466                  4
4            5   60    Male  102468.096593                  6

   Purchase_Frequency  Avg_Purchase_Value  Customer_Satisfaction  \
0            8.399733          593.396084               0.636181
1            9.763495          815.160713               0.684061
2            3.755147          772.152883               0.912728
3            7.873615          196.204909               0.604003
4            7.289206          191.786996               0.921790

   Customer_Lifetime_Value
0             10624.205465
1             18699.997206
2             25273.231878
3              8521.552537
4              3777.425089
```

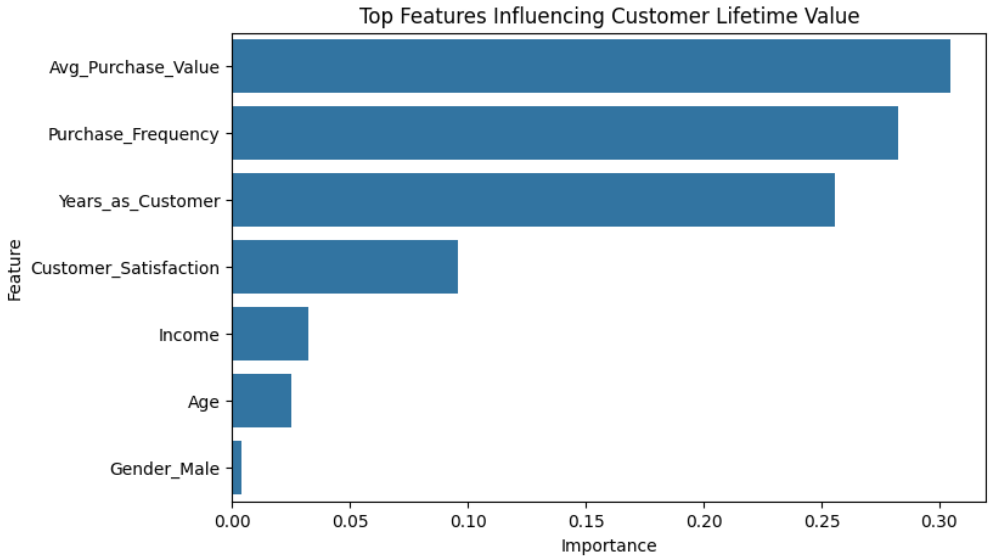








```
--- Model Evaluation ---
Mean Absolute Error (MAE): 4680.65
Mean Squared Error (MSE): 34722635.45
R-squared (R2): 0.824

Feature Importance:
                 Feature  Importance
4     Avg_Purchase_Value    0.304647
3     Purchase_Frequency    0.282515
2      Years_as_Customer    0.255952
5  Customer_Satisfaction    0.095838
1                 Income    0.032188
0                    Age    0.025097
6            Gender_Male    0.003762
```



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

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

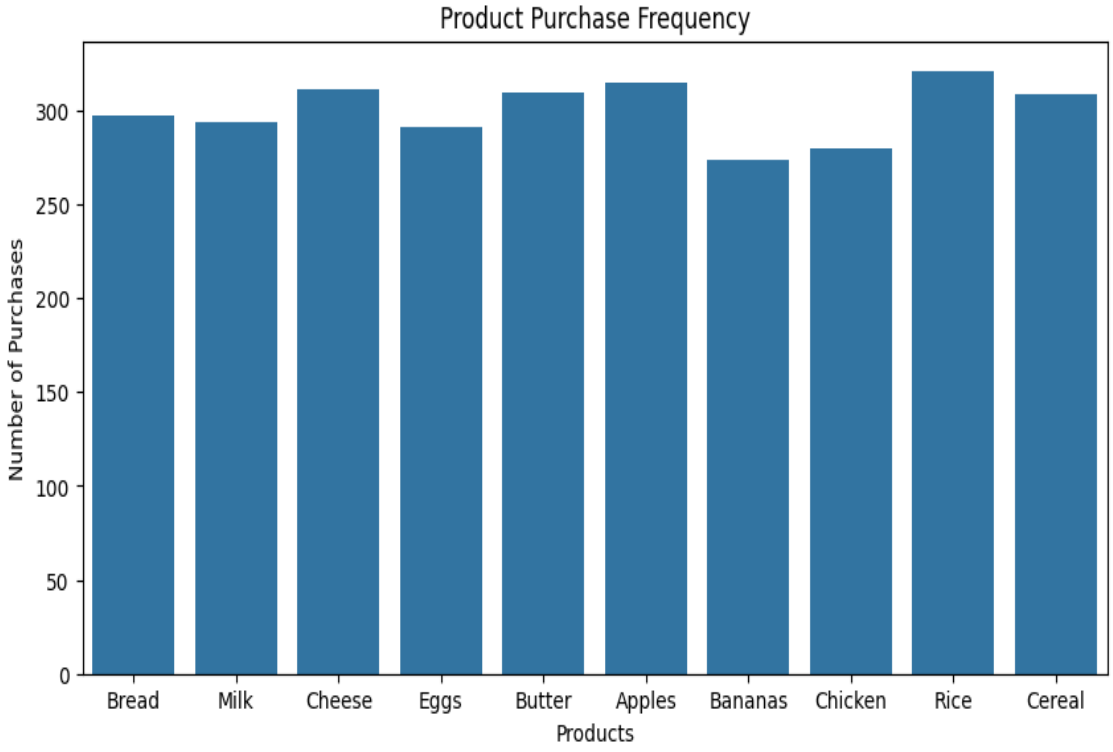
PROGRAM

```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.frequent_patterns import apriori, association_rules

# Step 2: Simulate Transactions Dataset
np.random.seed(42)
products = ['Bread', 'Milk', 'Cheese', 'Eggs', 'Butter',
            'Apples', 'Bananas', 'Chicken', 'Rice', 'Cereal']

transactions = []
for _ in range(1000):   # Simulate 1000 transactions
```

```
    transaction = np.random.choice(products, size=np.random.randint(1, 6), replace=False)
    transactions.append(transaction.tolist())

# Convert transactions to a one-hot encoded DataFrame
rows = []
for t in transactions:
    row = {product: (product in t) for product in products}
    rows.append(row)

all_products = pd.DataFrame(rows)

# Step 3: Data Exploration
print("First 5 transactions:")
print(all_products.head())
print("\nPurchase count for each product:")
print(all_products.sum())

# Step 4: Data Visualization
plt.figure(figsize=(10,5))
sns.barplot(x=all_products.sum().index, y=all_products.sum().values)
plt.title("Product Purchase Frequency")
plt.xlabel("Products")
plt.ylabel("Number of Purchases")
plt.show()

# Step 5 & 6: Model Selection and Training (Apriori)
frequent_itemsets = apriori(all_products, min_support=0.05, use_colnames=True)
print("\nTop Frequent Itemsets:")
print(frequent_itemsets.sort_values(by="support", ascending=False).head(10))

# Generate Association Rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules = rules.sort_values(by="lift", ascending=False)
print("\nTop Association Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head(10))

# Step 7: Model Evaluation / Insights
print("\nInsights for Cross-selling / Upselling:")
for index, row in rules.head(10).iterrows():
    print(f"If a customer buys {list(row['antecedents'])}, "
          f"they are likely to buy {list(row['consequents'])} "
          f"with confidence {row['confidence']:.2f} and lift {row['lift']:.2f}")
```

OUTPUT

```
First 5 transactions:
   Bread   Milk  Cheese   Eggs  Butter  Apples  Bananas  Chicken   Rice  \
0   True  False   False   True   False   False    False    False   True
1   True  False    True  False   False   False    False    False   True
2  False   True    True  False   False   False     True     True  False
3  False   True   False  False   False   False    False    False   True
4  False  False    True  False   False   False    False    False  False

   Cereal
0    True
1   False
2   False
3   False
4   False

Purchase count for each product:
Bread      297
Milk       294
Cheese     311
Eggs       291
Butter     310
Apples     315
Bananas    274
Chicken    280
Rice       321
Cereal     309
dtype: int64
```



Top Frequent Itemsets:

	support	itemsets
8	0.321	(Rice)
5	0.315	(Apples)
2	0.311	(Cheese)
4	0.310	(Butter)
9	0.309	(Cereal)
0	0.297	(Bread)
1	0.294	(Milk)
3	0.291	(Eggs)
7	0.280	(Chicken)
6	0.274	(Bananas)

Top Association Rules:

	antecedents	consequents	support	confidence	lift
9	(Chicken)	(Bread)	0.093	0.332143	1.118326
8	(Bread)	(Chicken)	0.093	0.313131	1.118326
20	(Rice)	(Eggs)	0.104	0.323988	1.113359
21	(Eggs)	(Rice)	0.104	0.357388	1.113359
22	(Apples)	(Butter)	0.106	0.336508	1.085509
23	(Butter)	(Apples)	0.106	0.341935	1.085509
14	(Cheese)	(Apples)	0.106	0.340836	1.082019
15	(Apples)	(Cheese)	0.106	0.336508	1.082019
18	(Apples)	(Eggs)	0.099	0.314286	1.080020
19	(Eggs)	(Apples)	0.099	0.340206	1.080020

Insights for Cross-selling / Upselling:
If a customer buys ['Chicken'], they are likely to buy ['Bread'] with confidence 0.33 and lift 1.12
If a customer buys ['Bread'], they are likely to buy ['Chicken'] with confidence 0.31 and lift 1.12
If a customer buys ['Rice'], they are likely to buy ['Eggs'] with confidence 0.32 and lift 1.11
If a customer buys ['Eggs'], they are likely to buy ['Rice'] with confidence 0.36 and lift 1.11
If a customer buys ['Apples'], they are likely to buy ['Butter'] with confidence 0.34 and lift 1.09
If a customer buys ['Butter'], they are likely to buy ['Apples'] with confidence 0.34 and lift 1.09
If a customer buys ['Cheese'], they are likely to buy ['Apples'] with confidence 0.34 and lift 1.08
If a customer buys ['Apples'], they are likely to buy ['Cheese'] with confidence 0.34 and lift 1.08
If a customer buys ['Apples'], they are likely to buy ['Eggs'] with confidence 0.31 and lift 1.08
If a customer buys ['Eggs'], they are likely to buy ['Apples'] with confidence 0.34 and lift 1.08

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

RESULT

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

Overall Record Completion Status	
Completed	
Date of Completion	
Faculty Signature	