The Apriori algorithm is used in unsupervised learning for mining frequent itemsets and learning association rules over transactional databases. It is particularly useful for identifying relationships between variables in large datasets.

Purpose of Apriori Algorithm

The primary purpose of the Apriori algorithm is to find associations between different sets of items in large datasets. This is often used in market basket analysis to find patterns in consumer purchasing behavior, but it can be applied to any domain where relationships between items need to be understood.

Suitable Type of Data

The Apriori algorithm is most suitable for categorical data, specifically transactional data where each transaction is a set of items. Examples of suitable data include:

- Retail transaction data
- Web clickstream data
- Biological data (e.g., gene sequences)
- Social network data

Examples

1. Market Basket Analysis

In a supermarket, the Apriori algorithm can be used to find associations between products bought together by customers. For example, it might find that customers who buy bread also frequently buy butter.

2. Web Usage Mining

In web analytics, the algorithm can identify patterns in user behavior. For example, it might find that users who visit a sports news page often also visit a page about health and fitness.

3. Medical Diagnosis

In healthcare, the algorithm can help find associations between different symptoms and diseases. For instance, it might find that patients who have a fever and cough also often have a sore throat.

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from mlxtend.frequent_patterns import apriori, association_rules
```

```
# Load dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx'
df = pd.read_excel(url)

# Preprocessing
df = df[df['Quantity'] > 0]  # Remove negative quantities
df = df.dropna(subset=['InvoiceNo', 'StockCode', 'Description'])  # Drop rows with missing values

# Create a binary matrix
basket = (df[df['Country'] == 'United Kingdom']
        .groupby(['InvoiceNo', 'Description'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))

def encode_units(x):
    return 1 if x >= 1 else 0

basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)  # Remove "POSTAGE" column

# Apply Apriori algorithm
frequent_itemsets = apriori(basket_sets, min_support=0.02, use_colnames=True)

# Generate association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)

# Visualize frequent itemsets
frequent_itemsets.sort_values(by='support', ascending=False).head(10).plot(
```

```
    kind='bar', x='itemsets', y='support', legend=False)
plt.title('Top 10 Frequent Itemsets')
plt.ylabel('Support')
plt.xticks(rotation=45)
plt.show()


# Visualize association rules
plt.figure(figsize=(10, 6))
sns.scatterplot(x="support", y="confidence", size="lift", data=rules)
plt.title('Association Rules')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.show()
```

2:

The Eclat algorithm (Equivalence Class Clustering and bottom-up Lattice Traversal) is a popular algorithm used in data mining for frequent itemset mining. It is an efficient algorithm for finding frequent itemsets in a transaction database, which can then be used to generate association rules. The primary purpose of the Eclat algorithm is to discover frequent itemsets in a dataset, which is a fundamental step in association rule learning.

Purpose of Eclat Algorithm

The Eclat algorithm is used for:

1. **Market Basket Analysis**: Identifying products that frequently co-occur in transactions.
2. **Recommendation Systems**: Recommending items to users based on frequently occurring itemsets.
3. **Intrusion Detection**: Identifying frequently occurring patterns in network traffic that may indicate security threats.
4. **Bioinformatics**: Finding frequent patterns in biological data, such as gene sequences.

Suitable Data for Eclat Algorithm

The Eclat algorithm is suitable for transactional data, where the data consists of a set of transactions, and each transaction is a set of items. Examples of such data include:

- Retail transactions (market basket data)

- Web usage logs
- Medical records
- Text data (e.g., frequent words in documents)

Implementation

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from mlxtend.frequent_patterns import apriori, association_rules


# Load the dataset

url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/00352/Online%20Retail.xlsx'

df = pd.read_excel(url)


# Display the first few rows

print("First few rows of the dataset:\n", df.head())


# Visualize the distribution of the number of items per transaction

df['InvoiceNo'] = df['InvoiceNo'].astype(str)

transactions = df.groupby('InvoiceNo')['StockCode'].count()

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

plt.hist(transactions, bins=50, color='skyblue')

plt.xlabel('Number of Items per Transaction')

plt.ylabel('Frequency')

plt.title('Distribution of Number of Items per Transaction')

plt.show()


# Filter out canceled transactions
```

```
df = df[~df['InvoiceNo'].str.startswith('C')]


# Create a basket (one-hot encoded) DataFrame

basket = df.groupby(['InvoiceNo',
'Description'])['Quantity'].sum().unstack().reset_index().fillna(0).set_index('InvoiceNo')


# Convert quantities to 1 (presence) and 0 (absence)

basket = basket.applymap(lambda x: 1 if x >= 1 else 0)


# Display the basket

print("Basket (one-hot encoded) DataFrame:\n", basket.head())


# Apply Apriori algorithm

frequent_itemsets = apriori(basket, min_support=0.01, use_colnames=True)

print("Frequent Itemsets:\n", frequent_itemsets.head())


# Generate association rules

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

print("\nAssociation Rules:\n", rules.head())


# Visualize frequent itemsets

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

sns.barplot(x='support', y='itemsets', data=frequent_itemsets.sort_values(by='support',
ascending=False).head(10))

plt.xlabel('Support')

plt.ylabel('Itemsets')

plt.title('Top 10 Frequent Itemsets')

plt.show()
```

```
# Visualize association rules
rules['antecedents'] = rules['antecedents'].apply(lambda x: ', '.join(list(x)))
rules['consequents'] = rules['consequents'].apply(lambda x: ', '.join(list(x)))

plt.figure(figsize=(10, 6))
sns.scatterplot(x='support', y='confidence', size='lift', data=rules, legend=False)
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Association Rules (Support vs Confidence)')
plt.show()
```