Installation

First, ensure you have the necessary libraries installed:

```
bash
Copy code
pip install numpy pandas scikit-learn matplotlib seaborn
```

1. K-Means Clustering

```
python
Copy code
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

# Load dataset
iris = load_iris()
X = iris.data

# K-Means Clustering
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
predictions = kmeans.predict(X)

# Plotting
plt.scatter(X[:, 0], X[:, 1], c=predictions)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300,
c='red')
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

2. Hierarchical Clustering

```
python
Copy code
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from scipy.cluster.hierarchy import dendrogram, linkage

# Load dataset
iris = load_iris()
X = iris.data

# Hierarchical Clustering
Z = linkage(X, 'ward')

# Dendrogram
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Samples')
plt.ylabel('Distance')
plt.show()
```

3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

```
python
Copy code
from sklearn.datasets import load_iris
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data

# DBSCAN Clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
predictions = dbscan.fit_predict(X)

# Plotting
plt.scatter(X[:, 0], X[:, 1], c=predictions)
plt.title('DBSCAN Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

4. Principal Component Analysis (PCA)

```
python
Copy code
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data

# PCA
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)

# Plotting
plt.scatter(X_reduced[:, 0], X_reduced[:, 1])
plt.title('PCA of Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

5. t-Distributed Stochastic Neighbor Embedding (t-SNE)

```
python
Copy code
from sklearn.datasets import load_iris
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data

# t-SNE
tsne = TSNE(n_components=2)
X_embedded = tsne.fit_transform(X)
```

```
# Plotting
plt.scatter(X_embedded[:, 0], X_embedded[:, 1])
plt.title('t-SNE of Iris Dataset')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.show()
```

6. Gaussian Mixture Models (GMM)

```
python
Copy code
from sklearn.datasets import load_iris
from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data

# GMM
gmm = GaussianMixture(n_components=3)
gmm.fit(X)
predictions = gmm.predict(X)

# Plotting
plt.scatter(X[:, 0], X[:, 1], c=predictions)
plt.title('GMM Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

7. Autoencoders

```
python
Copy code
import numpy as np
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers

# Load dataset
iris = load_iris()
X = iris.data
X_scaled = StandardScaler().fit_transform(X)

# Autoencoder model
input_dim = X_scaled.shape[1]
encoding_dim = 2

autoencoder = keras.Sequential([
    layers.Input(shape=(input_dim,)),
    layers.Dense(encoding_dim, activation='relu'),
    layers.Dense(input_dim, activation='sigmoid')
])

autoencoder.compile(optimizer='adam', loss='mean_squared_error')
autoencoder.fit(X_scaled, X_scaled, epochs=100, batch_size=10, shuffle=True)

# Encoding
encoder = keras.Sequential([
    layers.Input(shape=(input_dim,)),
    layers.Dense(encoding_dim, activation='relu')
])

encoded_data = encoder.predict(X_scaled)

# Plotting
plt.scatter(encoded_data[:, 0], encoded_data[:, 1])
plt.title('Autoencoder Encoded Data')
plt.xlabel('Encoded Feature 1')
plt.ylabel('Encoded Feature 2')
plt.show()
```

8. Association Rule Learning (Apriori)

For association rule learning, we typically use a transactional dataset. Here's an example using the `mlxtend` library for the Apriori algorithm.

Install `mlxtend`:

```
bash
Copy code
pip install mlxtend
```

Apriori Example:

```
python
Copy code
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

# Sample transactional data
data = {'Transaction': [1, 1, 1, 2, 2, 3, 3, 3, 4, 4],
        'Item': ['Bread', 'Milk', 'Eggs', 'Bread', 'Diaper',
                 'Milk', 'Diaper', 'Beer', 'Milk', 'Diaper']}

df = pd.DataFrame(data)
basket = df.groupby(['Transaction', 'Item'])
['Item'].count().unstack().reset_index().fillna(0).set_index('Transaction')

# Convert to 1s and 0s
basket = basket.applymap(lambda x: 1 if x > 0 else 0)

# Apriori
frequent_itemsets = apriori(basket, min_support=0.4, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

print(rules)
```