Unsupervised Learning Projects

Creating practice projects for unsupervised learning is a great way to get hands-on experience with machine learning techniques such as clustering and dimensionality reduction. Here's a working list of projects, datasets, and detailed instructions, including a full pipeline for one project.

Project Ideas and Datasets

1. **Customer Segmentation**
   - **Dataset**: Online Retail Dataset
   - **Description**: This dataset contains transactional data of an online retail store. The task is to segment customers based on their purchasing behavior.
2. **Image Compression**
   - **Dataset**: CIFAR-10 Dataset
   - **Description**: Use k-means clustering to compress images by reducing the number of colors.
3. **Anomaly Detection in Network Traffic**
   - **Dataset**: KDD Cup 1999 Data
   - **Description**: Identify anomalies in network traffic data.
4. **Topic Modeling of News Articles**
   - **Dataset**: 20 Newsgroups Dataset
   - **Description**: Apply topic modeling to categorize news articles into topics.
5. **Dimensionality Reduction for Visualization**
   - **Dataset**: Wine Dataset
   - **Description**: Use PCA or t-SNE to reduce dimensions for visualization.

Step-by-Step Guide with Python Pipeline

Let's create a detailed step-by-step guide for the **Customer Segmentation** project.

Step 1: Data Preparation

1. **Download the Dataset**:
   - Visit the Online Retail Dataset page.
   - Download and extract the data.
2. **Load the Data**:

```python
import pandas as pd

# Load the dataset
data = pd.read_excel('Online_Retail.xlsx')
```

```python
# Display the first few rows
print(data.head())
```

3. **Data Cleaning**:
   - Remove rows with missing values.
   - Filter out returns and transactions without valid customer IDs.

```python
# Remove missing values
data.dropna(inplace=True)

# Filter out negative quantities (returns)
data = data[data['Quantity'] > 0]

# Filter out rows without CustomerID
data = data[data['CustomerID'].notnull()]

print(data.shape)
```

Step 2: Feature Engineering

1. **Calculate Total Spend**:

```python
# Create a new column for total spend
data['TotalSpend'] = data['Quantity'] * data['UnitPrice']
```

2. **Aggregate Data by Customer**:

```python
# Aggregate data to calculate metrics for each customer
customer_data = data.groupby('CustomerID').agg({
    'InvoiceNo': 'nunique',  # Number of transactions
    'Quantity': 'sum',       # Total quantity purchased
    'TotalSpend': 'sum'      # Total spend
}).rename(columns={
    'InvoiceNo': 'NumTransactions',
    'Quantity': 'TotalQuantity'
})

print(customer_data.head())
```

Step 3: Data Preprocessing

1. **Normalize Features**:

```python
from sklearn.preprocessing import StandardScaler

# Normalize the features
scaler = StandardScaler()
```

```
customer_data_scaled = scaler.fit_transform(customer_data)

print(customer_data_scaled[:5])
```

Step 4: Clustering

1. **Apply K-Means Clustering**:

```python
from sklearn.cluster import KMeans

# Apply K-Means clustering
kmeans = KMeans(n_clusters=4, random_state=42)
customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)

print(customer_data.head())
```

2. **Visualize Clusters**:

```python
import matplotlib.pyplot as plt

# Visualize the clusters
plt.scatter(customer_data['TotalSpend'], customer_data['NumTransactions'],
c=customer_data['Cluster'])
plt.xlabel('Total Spend')
plt.ylabel('Number of Transactions')
plt.title('Customer Segments')
plt.show()
```

Step 5: Evaluation

1. **Analyze Cluster Characteristics**:

```python
# Analyze characteristics of each cluster
cluster_analysis = customer_data.groupby('Cluster').mean()
print(cluster_analysis)
```

Full Pipeline

Here's the full code for the pipeline:

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the dataset
```

```
    data = pd.read_excel('Online_Retail.xlsx')

    # Data cleaning
    data.dropna(inplace=True)
    data = data[data['Quantity'] > 0]
    data = data[data['CustomerID'].notnull()]

    # Feature engineering
    data['TotalSpend'] = data['Quantity'] * data['UnitPrice']
    customer_data = data.groupby('CustomerID').agg({
        'InvoiceNo': 'nunique',  # Number of transactions
        'Quantity': 'sum',       # Total quantity purchased
        'TotalSpend': 'sum'      # Total spend
    }).rename(columns={
        'InvoiceNo': 'NumTransactions',
        'Quantity': 'TotalQuantity'
    })

    # Data preprocessing
    scaler = StandardScaler()
    customer_data_scaled = scaler.fit_transform(customer_data)

    # K-Means clustering
    kmeans = KMeans(n_clusters=4, random_state=42)
    customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)

    # Visualize clusters
    plt.scatter(customer_data['TotalSpend'], customer_data['NumTransactions'],
    c=customer_data['Cluster'])
    plt.xlabel('Total Spend')
    plt.ylabel('Number of Transactions')
    plt.title('Customer Segments')
    plt.show()

    # Cluster analysis
    cluster_analysis = customer_data.groupby('Cluster').mean()
    print(cluster_analysis)
```

Answer Key

Key Steps in the Pipeline:

- **Data Cleaning**: Ensures data quality by removing irrelevant rows.
- **Feature Engineering**: Calculates meaningful features for clustering.
- **Normalization**: Standardizes features for better clustering performance.
- **Clustering**: Groups customers based on similar behaviors.
- **Visualization and Analysis**: Provides insights into each cluster's characteristics.

This project demonstrates the full cycle of data preparation, feature engineering, unsupervised learning, and evaluation. You can adapt these steps to other datasets and projects in unsupervised learning. Let me know if you need further details or additional projects!

You're right! Simply dropping rows with missing data can result in loss of valuable information. Instead of dropping rows, we can employ several strategies to handle missing data more effectively. Here, I'll update the project to address missing values through imputation or other relevant techniques.

Updated Data Cleaning with Imputation

Step 1: Data Preparation

1. **Download the Dataset**:
    - Visit the Online Retail Dataset page.
    - Download and extract the data.
2. **Load the Data**:

```python
import pandas as pd

# Load the dataset
data = pd.read_excel('Online_Retail.xlsx')

# Display the first few rows
print(data.head())
```

3. **Data Cleaning with Imputation**:
    - Use median imputation for missing numeric data.
    - Use the mode for categorical features like `CustomerID` and `Country`.

```python
# Remove negative quantities (returns)
data = data[data['Quantity'] > 0]

# Fill missing CustomerID with the most frequent value (mode)
most_frequent_customer = data['CustomerID'].mode()[0]
data['CustomerID'].fillna(most_frequent_customer, inplace=True)

# Fill missing UnitPrice with the median price
median_unit_price = data['UnitPrice'].median()
data['UnitPrice'].fillna(median_unit_price, inplace=True)

# Display the number of missing values to ensure no missing data remains
print(data.isnull().sum())
```

Step 2: Feature Engineering

1. **Calculate Total Spend**:

```python
# Create a new column for total spend
data['TotalSpend'] = data['Quantity'] * data['UnitPrice']
```

2. **Aggregate Data by Customer**:

```python
# Aggregate data to calculate metrics for each customer
customer_data = data.groupby('CustomerID').agg({
    'InvoiceNo': 'nunique',  # Number of transactions
    'Quantity': 'sum',       # Total quantity purchased
    'TotalSpend': 'sum'      # Total spend
```

```
}).rename(columns={
    'InvoiceNo': 'NumTransactions',
    'Quantity': 'TotalQuantity'
})

print(customer_data.head())
```

Step 3: Data Preprocessing

1. **Normalize Features**:

```python
from sklearn.preprocessing import StandardScaler

# Normalize the features
scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customer_data)

print(customer_data_scaled[:5])
```

Step 4: Clustering

1. **Apply K-Means Clustering**:

```python
from sklearn.cluster import KMeans

# Apply K-Means clustering
kmeans = KMeans(n_clusters=4, random_state=42)
customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)

print(customer_data.head())
```

2. **Visualize Clusters**:

```python
import matplotlib.pyplot as plt

# Visualize the clusters
plt.scatter(customer_data['TotalSpend'], customer_data['NumTransactions'],
c=customer_data['Cluster'])
plt.xlabel('Total Spend')
plt.ylabel('Number of Transactions')
plt.title('Customer Segments')
plt.show()
```

Step 5: Evaluation

1. **Analyze Cluster Characteristics**:

```python
# Analyze characteristics of each cluster
cluster_analysis = customer_data.groupby('Cluster').mean()
print(cluster_analysis)
```

Full Pipeline with Imputation

Here's the complete pipeline with improved data cleaning using imputation:

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_excel('Online_Retail.xlsx')

# Data cleaning with imputation
data = data[data['Quantity'] > 0]
most_frequent_customer = data['CustomerID'].mode()[0]
data['CustomerID'].fillna(most_frequent_customer, inplace=True)
median_unit_price = data['UnitPrice'].median()
data['UnitPrice'].fillna(median_unit_price, inplace=True)

# Feature engineering
data['TotalSpend'] = data['Quantity'] * data['UnitPrice']
customer_data = data.groupby('CustomerID').agg({
    'InvoiceNo': 'nunique',  # Number of transactions
    'Quantity': 'sum',       # Total quantity purchased
    'TotalSpend': 'sum'      # Total spend
}).rename(columns={
    'InvoiceNo': 'NumTransactions',
    'Quantity': 'TotalQuantity'
})

# Data preprocessing
scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customer_data)

# K-Means clustering
kmeans = KMeans(n_clusters=4, random_state=42)
customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)

# Visualize clusters
plt.scatter(customer_data['TotalSpend'], customer_data['NumTransactions'],
c=customer_data['Cluster'])
plt.xlabel('Total Spend')
plt.ylabel('Number of Transactions')
plt.title('Customer Segments')
plt.show()

# Cluster analysis
cluster_analysis = customer_data.groupby('Cluster').mean()
print(cluster_analysis)
```

Key Changes

- **Imputation**: We filled missing values in the **`CustomerID`** column with the most frequent value, and missing **`UnitPrice`** values with the median price. This approach helps preserve data integrity while addressing missing values efficiently.

This updated pipeline ensures that we don't lose valuable data and can still make meaningful segmentations based on available information. Let me know if there's anything else you'd like to explore!