Aim

To perform customer segmentation using K-Means clustering on the Online Retail dataset and identify outliers using various methods.

Algorithm

1. Load and preprocess the dataset (handle missing values, filter out invalid entries).
2. Encode categorical variables and scale numerical features.
3. Determine the optimal number of clusters (K) using the Elbow method and Silhouette score.
4. Apply K-Means clustering and assign labels to data points.
5. Detect outliers using distance from cluster centers, Z-score method, and Isolation Forest.
6. Visualize clustering results using PCA.

Algorithm Description

K-Means is an unsupervised clustering algorithm that partitions data into K clusters by minimizing intra-cluster variance. The algorithm iteratively:

- Assigns each data point to the nearest cluster center.
- Updates cluster centers by computing the mean of assigned points.
- Repeats until convergence.

Outlier detection methods include:

- **Distance-based outliers**: Identifies data points far from cluster centers.
- **Z-Score method**: Flags points with values beyond three standard deviations.
- **Isolation Forest**: Detects anomalies based on recursive partitioning of data.

Results

- The optimal number of clusters (K) was determined using the silhouette score.
- Clustering results were visualized using PCA-reduced data.
- Outliers were successfully detected using multiple methods.
- The model provides customer segmentation insights based on purchase behavior.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score, davies_bouldin_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import pairwise_distances_argmin_min
from scipy.stats import zscore
from sklearn.ensemble import IsolationForest

df = pd.read_excel('Online Retail.xlsx')
df
```

```
       InvoiceNo StockCode                          Description
Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6
1         536365     71053                  WHITE METAL LANTERN
6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER
8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE
6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.
6
...          ...       ...                                  ...
...
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS
12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL
6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL
4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE
4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT
3
```

```
               InvoiceDate  UnitPrice  CustomerID         Country
0      2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2      2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
...                    ...        ...         ...             ...
541904 2011-12-09 12:50:00       0.85     12680.0          France
541905 2011-12-09 12:50:00       2.10     12680.0          France
541906 2011-12-09 12:50:00       4.15     12680.0          France
541907 2011-12-09 12:50:00       4.15     12680.0          France
541908 2011-12-09 12:50:00       4.95     12680.0          France

[541909 rows x 8 columns]
```

```
df.shape
```

```
(541909, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
df.isna().sum()
```

```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

```python
df['InvoiceDate'] = df['InvoiceDate'].apply(lambda x: str(x).split()
[0].split('-')[0])
df
```

```
       InvoiceNo StockCode                          Description
Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6
1         536365     71053                  WHITE METAL LANTERN
6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER
8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE
6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.
6
...          ...       ...                                  ...
...
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS
12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL 
6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL 
4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE
4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT 
3

       InvoiceDate  UnitPrice  CustomerID         Country
0             2010       2.55     17850.0  United Kingdom
1             2010       3.39     17850.0  United Kingdom
2             2010       2.75     17850.0  United Kingdom
3             2010       3.39     17850.0  United Kingdom
4             2010       3.39     17850.0  United Kingdom
...            ...        ...         ...             ...
541904        2011       0.85     12680.0          France
541905        2011       2.10     12680.0          France
541906        2011       4.15     12680.0          France
541907        2011       4.15     12680.0          France
541908        2011       4.95     12680.0          France

[541909 rows x 8 columns]
```

```python
df = df.dropna(subset=['Description'])
df.shape
```

```
(540455, 8)
```

```python
print(f"Missing values after cleaning: {df.isna().sum().sum()}")
```

```
Missing values after cleaning: 133626
```

```
imputer_num = SimpleImputer(strategy='median')
df.loc[:, 'CustomerID'] =
imputer_num.fit_transform(df[['CustomerID']])
```

```
df = df[df['Quantity'] > 0]
df = df[df['UnitPrice'] > 0]
```

```
df.shape
```

```
(530104, 8)
```

```
df.isna().sum().sum()
```

```
0
```

```
label_encoder = LabelEncoder()
```

```
for col in df.columns:

    if df[col].dtype == 'object':
        df[col] = df[col].astype(str)
        df[col] = label_encoder.fit_transform(df[col])
```

```
df
```

```
        InvoiceNo  StockCode  Description  Quantity  InvoiceDate
UnitPrice  \
0               0       3407         3844         6            0
2.55
1               0       2729         3852         6            0
3.39
2               0       2953          888         8            0
2.75
3               0       2897         1859         6            0
3.39
4               0       2896         2849         6            0
3.39
...           ...        ...          ...       ...          ...
...
541904      19958       1489         2321        12            1
0.85
541905      19958       1765          718         6            1
2.10
541906      19958       2105          724         4            1
4.15
541907      19958       2106          723         4            1
4.15
541908      19958       1056          282         3            1
4.95
```

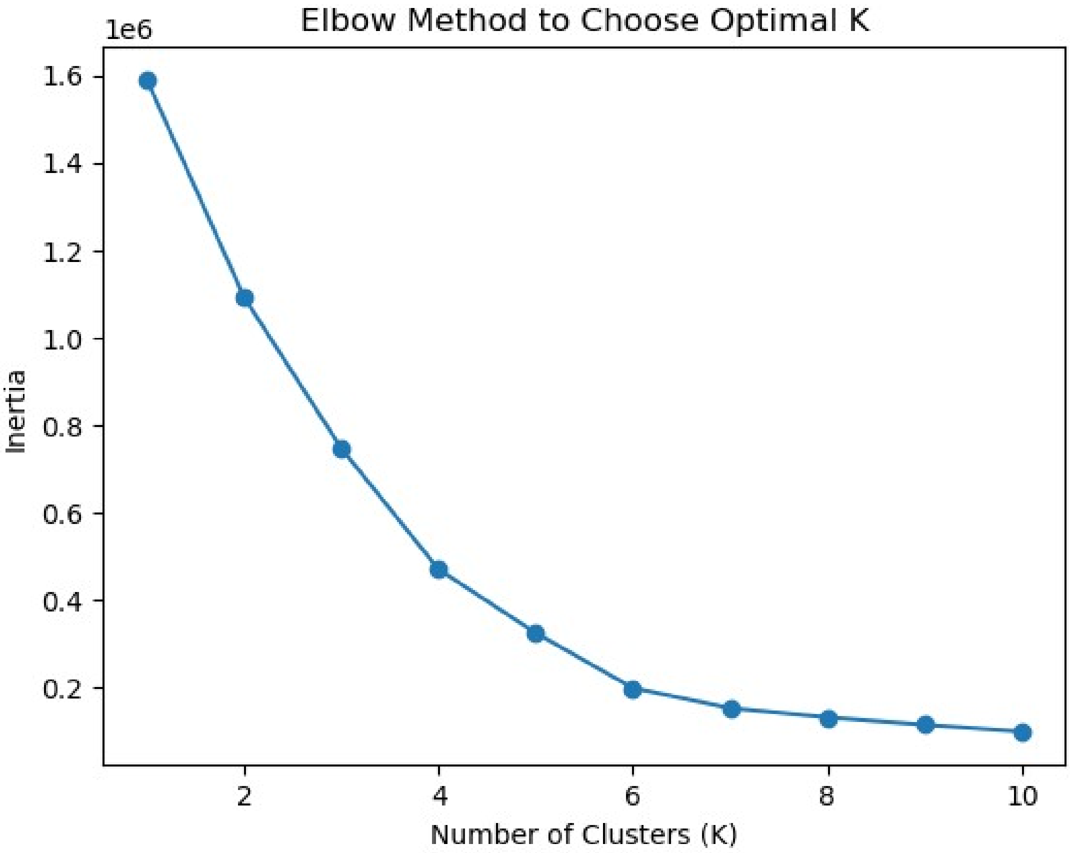
```
        CustomerID  Country
0          17850.0       36
1          17850.0       36
2          17850.0       36
3          17850.0       36
4          17850.0       36
...            ...      ...
541904     12680.0       13
541905     12680.0       13
541906     12680.0       13
541907     12680.0       13
541908     12680.0       13

[530104 rows x 8 columns]
```

```
features = df[['Quantity', 'UnitPrice', 'CustomerID']]
scaler = StandardScaler()
scaled_data = scaler.fit_transform(features)

inertia = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)

plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method to Choose Optimal K')
plt.show()
```



```python
silhouette_scores = []
for k in k_range[1:]:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    kmeans.fit(scaled_data)
    score = silhouette_score(scaled_data, kmeans.labels_)
    silhouette_scores.append(score)

plt.plot(k_range[1:], silhouette_scores, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score for Different K Values')
plt.show()

optimal_k = k_range[1:][np.argmax(silhouette_scores)]
print(f"Optimal number of clusters (K) based on silhouette score:
{optimal_k}")

param_grid = {
    'n_clusters': [optimal_k, 4, 6],
    'n_init': [10],
    'init': ['k-means++']
}
```

```
best_silhouette_score = -1
best_kmeans = None
best_params = None

for n_clusters in param_grid['n_clusters']:
    for init in param_grid['init']:
        kmeans = KMeans(n_clusters=n_clusters,
n_init=param_grid['n_init'][0], init=init, random_state=42)
        kmeans.fit(scaled_data)
        score = silhouette_score(scaled_data, kmeans.labels_)
        if score > best_silhouette_score:
            best_silhouette_score = score
            best_kmeans = kmeans
            best_params = {
                'n_clusters': n_clusters,
                'init': init
            }

df['Cluster'] = best_kmeans.labels_

distances = pairwise_distances_argmin_min(scaled_data,
best_kmeans.cluster_centers_)[1]

threshold = np.percentile(distances, 95)
df['Outlier'] = np.where(distances > threshold, 1, 0)

pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_data)

plt.figure(figsize=(8, 6))
plt.scatter(pca_components[:, 0], pca_components[:, 1],
c=df['Cluster'], cmap='viridis', marker='o', label='Clusters')
plt.scatter(pca_components[df['Outlier'] == 1, 0],
pca_components[df['Outlier'] == 1, 1], color='red', label='Outliers',
marker='x')
plt.title(f'K-means Clustering with K={optimal_k} and Outliers')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.show()

print(f"Cluster Centers:\n{kmeans.cluster_centers_}")

df['Z_Score_Quantity'] = np.abs(zscore(df['Quantity']))
df['Z_Score_Price'] = np.abs(zscore(df['UnitPrice']))
df['Outlier_ZScore'] = np.where((df['Z_Score_Quantity'] > 3) |
(df['Z_Score_Price'] > 3), 1, 0)

iso_forest = IsolationForest(contamination=0.05, random_state=42)
df['Outlier_IsoForest'] = iso_forest.fit_predict(scaled_data)
```

```python
df['Outlier_IsoForest'] = df['Outlier_IsoForest'].apply(lambda x: 1 if x == -1 else 0)

outliers_tuned_df = df[df['Outlier'] == 1]
outliers_zscore_df = df[df['Outlier_ZScore'] == 1]
outliers_iso_df = df[df['Outlier_IsoForest'] == 1]

print("Outliers detected after KMeans:")
print(outliers_tuned_df)
print("\nOutliers detected using Z-Score method:")
print(outliers_zscore_df)
print("\nOutliers detected using Isolation Forest:")
print(outliers_iso_df)

print(f"Cluster Centers:\n{best_kmeans.cluster_centers_}")
```