In [2]:

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

In [3]:

```
df = pd.read_csv(r"C:\Users\vidhi\Downloads\Wine.csv")
df
```

Out[3]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavano
0	14.23	1.71	2.43	15.6	127	2.80	3
1	13.20	1.78	2.14	11.2	100	2.65	2
2	13.16	2.36	2.67	18.6	101	2.80	3
3	14.37	1.95	2.50	16.8	113	3.85	3
4	13.24	2.59	2.87	21.0	118	2.80	2
...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0
174	13.40	3.91	2.48	23.0	102	1.80	0
175	13.27	4.28	2.26	20.0	120	1.59	0
176	13.17	2.59	2.37	20.0	120	1.65	0
177	14.13	4.10	2.74	24.5	96	2.05	0

178 rows × 14 columns

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Alcohol                 0
Malic_Acid              0
Ash                     0
Ash_Alcanity            0
Magnesium               0
Total_Phenols           0
Flavanoids              0
Nonflavanoid_Phenols    0
Proanthocyanins         0
Color_Intensity         0
Hue                     0
OD280                   0
Proline                 0
Customer_Segment        0
dtype: int64
```

In [8]:

```
scaler = StandardScaler()
data = df.drop('Customer_Segment', axis=1)
x_fit = scaler.fit_transform(data)
x_fit = pd.DataFrame(x_fit, columns=data.columns)
pd.DataFrame(x_fit)
```

Out[8]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	F
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	
...	...	...	...	...	...	...	
173	0.876275	2.974543	0.305159	0.301803	-0.332922	-0.985614	
174	0.493343	1.412609	0.414820	1.052516	0.158572	-0.793334	
175	0.332758	1.744744	-0.389355	0.151661	1.422412	-1.129824	
176	0.209232	0.227694	0.012732	0.151661	1.422412	-1.033684	
177	1.395086	1.583165	1.365208	1.502943	-0.262708	-0.392751	

178 rows × 13 columns

In [10]:

```
y_fit = df['Customer_Segment']
```

In [21]:

```
df.columns
```

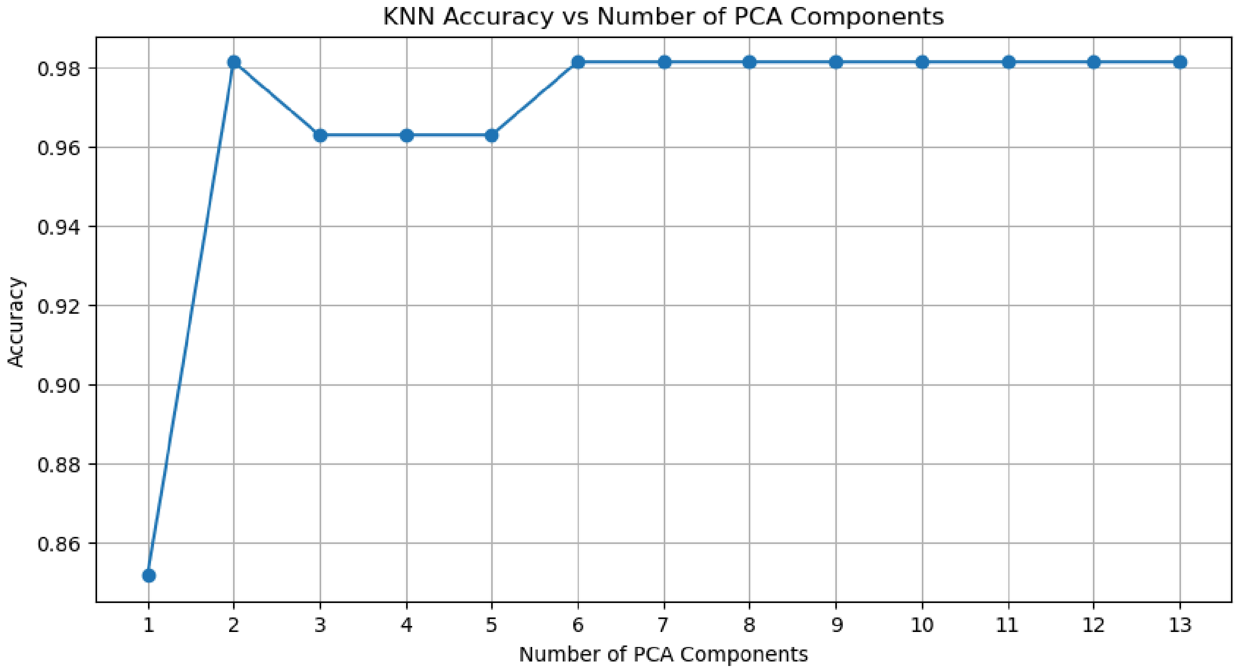
Out[21]:

```
Index(['Alcohol', 'Malic_Acid', 'Ash', 'Ash_Alcanity', 'Magnesium',
       'Total_Phenols', 'Flavanoids', 'Nonflavanoid_Phenols',
       'Proanthocyanins', 'Color_Intensity', 'Hue', 'OD280', 'Proline',
       'Customer_Segment'],
      dtype='object')
```

In [25]:

```
accuracies = []

# Loop through PCA components from 1 to 13
for n in range(1, 14):
    pca = PCA(n_components=n)
    X_pca = pca.fit_transform(x_fit)   # dataset now has only n features

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(
        X_pca, y_fit, test_size=0.3, random_state=42
    )

    # Train Logistic Regression
    logreg = LogisticRegression(max_iter=1000)
    logreg.fit(X_train, y_train)
    y_pred = logreg.predict(X_test)

    # Accuracy
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)

    print(f"PCA Components: {n}, Features used: {X_pca.shape[1]}, Accuracy: {a
# Plot
plt.figure(figsize=(10, 5))
plt.plot(range(1, 14), accuracies, marker='o')
plt.title('KNN Accuracy vs Number of PCA Components')
plt.xlabel('Number of PCA Components')
plt.ylabel('Accuracy')
plt.grid(True)
plt.xticks(range(1, 14))
plt.show()

# Best accuracy
best_n = accuracies.index(max(accuracies)) + 1
print(f"\n✅ Best accuracy: {max(accuracies):.4f} with {best_n} PCA components
```

```
PCA Components: 1, Features used: 1, Accuracy: 0.8519
PCA Components: 2, Features used: 2, Accuracy: 0.9815
PCA Components: 3, Features used: 3, Accuracy: 0.9630
PCA Components: 4, Features used: 4, Accuracy: 0.9630
PCA Components: 5, Features used: 5, Accuracy: 0.9630
PCA Components: 6, Features used: 6, Accuracy: 0.9815
PCA Components: 7, Features used: 7, Accuracy: 0.9815
PCA Components: 8, Features used: 8, Accuracy: 0.9815
PCA Components: 9, Features used: 9, Accuracy: 0.9815
PCA Components: 10, Features used: 10, Accuracy: 0.9815
PCA Components: 11, Features used: 11, Accuracy: 0.9815
PCA Components: 12, Features used: 12, Accuracy: 0.9815
PCA Components: 13, Features used: 13, Accuracy: 0.9815
```



```
✅ Best accuracy: 0.9815 with 2 PCA components
```

```
In [ ]:
```