In [1]:
```
import pandas as pd

url = "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

# Define column names (first is class label, then your 13 features)
col_names = ['Wine', 'Alcohol', 'Malic.acid', 'Ash', 'Alcalinity_of_ash', 'Mag
             'Total_phenols', 'Flavanoids', 'Nonflavanoid_phenols',
             'Proanthocyanins', 'Color_intensity', 'Hue',
             'OD280_OD315', 'Proline']

df = pd.read_csv(url, header=None, names=col_names)
```

In [3]:
```
df.head()
```

Out[3]:

	Wine	Alcohol	Malic.acid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols
0	1	14.23	1.71	2.43	15.6	127	2.80
1	1	13.20	1.78	2.14	11.2	100	2.65
2	1	13.16	2.36	2.67	18.6	101	2.80
3	1	14.37	1.95	2.50	16.8	113	3.85
4	1	13.24	2.59	2.87	21.0	118	2.80

In [5]:
```
df.dtypes
```

Out[5]:
```
Wine                    int64
Alcohol                 float64
Malic.acid              float64
Ash                     float64
Alcalinity_of_ash       float64
Magnesium                 int64
Total_phenols           float64
Flavanoids              float64
Nonflavanoid_phenols    float64
Proanthocyanins         float64
Color_intensity         float64
Hue                     float64
OD280_OD315             float64
Proline                   int64
dtype: object
```

In [7]:
```
df.size
```

Out[7]:
```
2492
```

In [9]:
```
df.shape
```

Out[9]:
```
(178, 14)
```

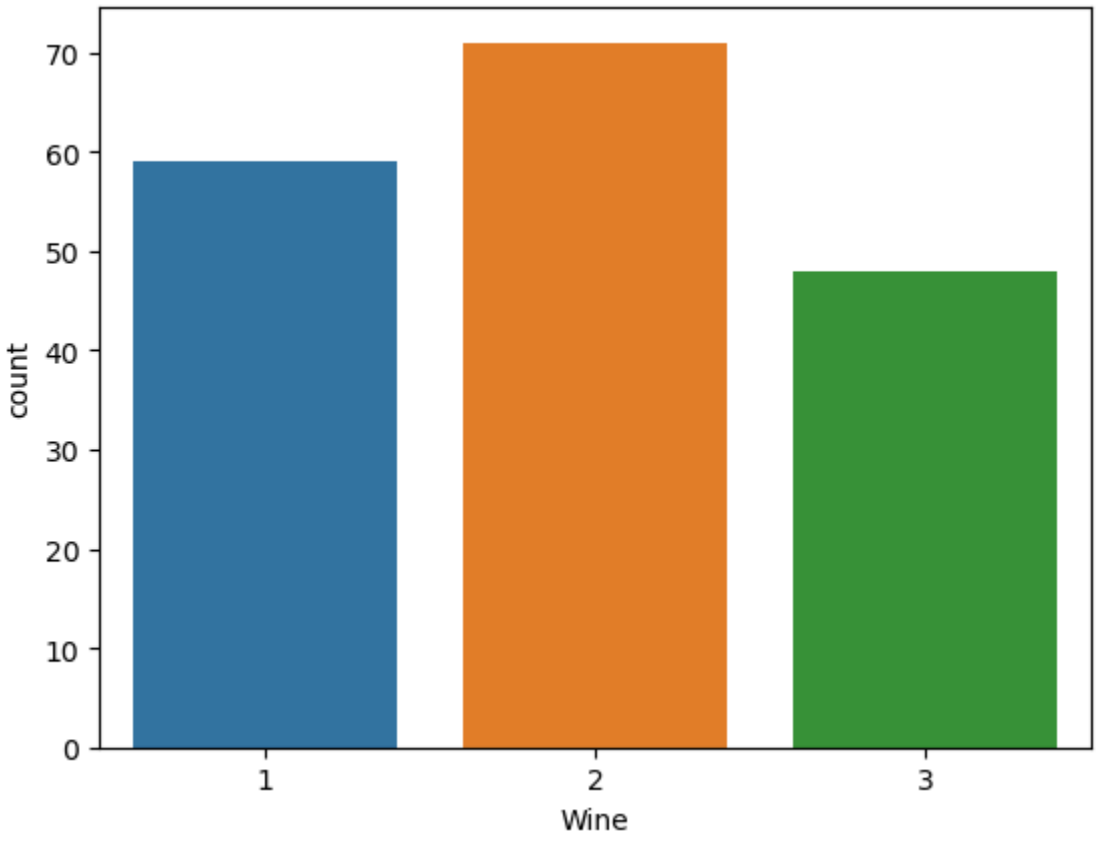
In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Wine                    0
Alcohol                 0
Malic.acid              0
Ash                     0
Alcalinity_of_ash       0
Magnesium               0
Total_phenols           0
Flavanoids              0
Nonflavanoid_phenols    0
Proanthocyanins         0
Color_intensity         0
Hue                     0
OD280_OD315             0
Proline                 0
dtype: int64
```

In [13]:
```
import seaborn as sns
```

In [15]:
```
sns.countplot(x = 'Wine',data=df)
```

Out[15]:
```
<Axes: xlabel='Wine', ylabel='count'>
```



In [17]:
```
target= df['Wine']
df = df.drop('Wine',axis=1)
```

In [19]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df,target,test_size =0.20,ran
```

```
X_train.head()
```

Out[19]:

	Alcohol	Malic.acid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flav
81	12.72	1.81	2.20	18.8	86	2.20	
6	14.39	1.87	2.45	14.6	96	2.50	
61	12.64	1.36	2.02	16.8	100	2.02	
126	12.43	1.53	2.29	21.5	86	2.74	
41	13.41	3.84	2.12	18.8	90	2.45	

In [21]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [23]:
```
X_train = pd.DataFrame(X_train)
X_test = pd.DataFrame(X_test)
```

In [25]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[25]:
```
▾ LogisticRegression

LogisticRegression()
```

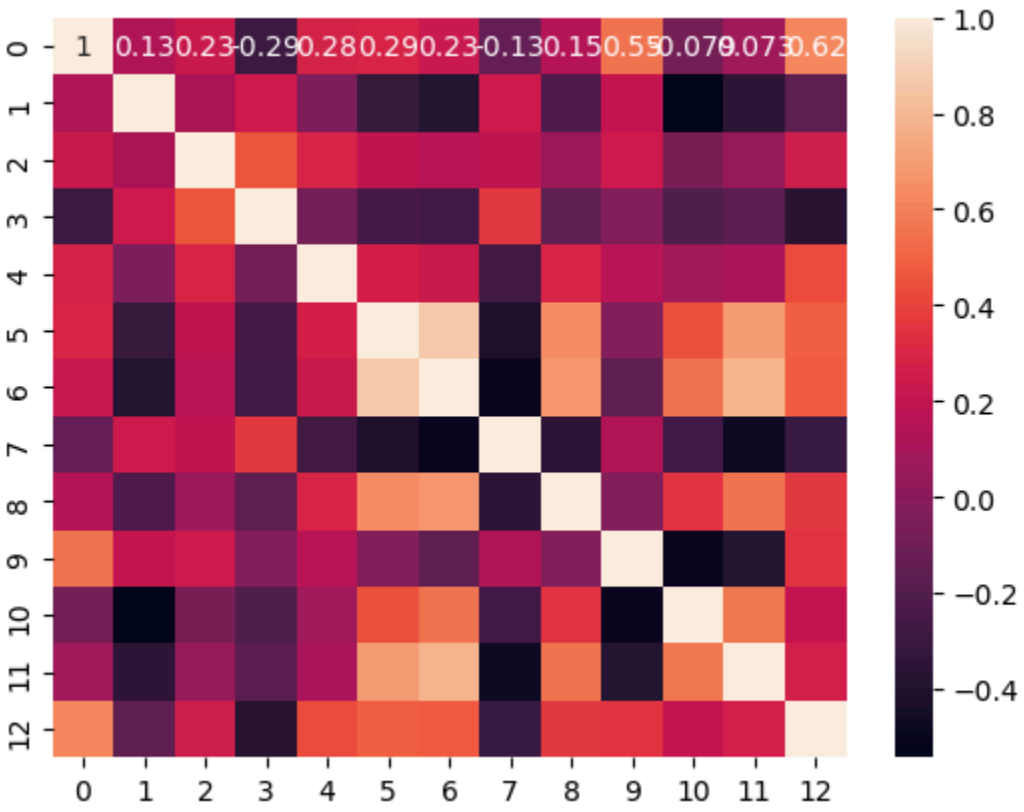
In [35]:
```
from sklearn.metrics import classification_report

y_predict= model.predict(X_test)
y_actual=y_test
print(classification_report(y_actual,y_predict))
y_predict
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        14
           2       1.00      0.91      0.95        11
           3       0.92      1.00      0.96        11

    accuracy                           0.97        36
   macro avg       0.97      0.97      0.97        36
weighted avg       0.97      0.97      0.97        36
```

Out[35]:
```
array([3, 3, 2, 1, 2, 3, 1, 3, 2, 3, 3, 3, 2, 3, 3, 1, 1, 1, 1, 3, 1, 1,
       2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 3, 1, 3])
```

In [39]:
```
sns.heatmap(X_train.corr(),annot=True)
```

Out[39]: <Axes: >



In [41]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
```

In [43]:
```
tr_comp = pca.fit_transform(X_train)
ts_comp = pca.fit_transform(X_test)
```

In [45]:
```
tr_comp
```

Out[45]:
```
array([[-9.60220620e-01,  1.37661234e+00],
       [-2.42530009e+00, -1.23017442e+00],
       [ 1.96733955e+00,  6.16496788e-01],
       [-9.51251080e-04,  1.15220462e+00],
       [-6.09309322e-01, -3.21338165e-01],
       [-2.56560022e+00, -1.80310461e+00],
       [-7.05068054e-01,  1.98167353e+00],
       [-2.00263605e+00, -1.66490389e+00],
       [-2.30867257e+00, -2.34471255e+00],
       [ 3.56879212e+00, -1.58250190e+00],
       [ 4.95893377e-01,  2.16611412e+00],
       [-2.57383431e+00, -1.32529656e+00],
       [ 1.82918771e+00,  9.21797684e-01],
       [ 2.43858969e+00, -2.43322759e+00],
       [-2.06697420e+00,  1.81955597e+00],
       [ 1.47618025e+00, -2.60164829e+00],
       [ 2.94629228e+00, -1.54181990e+00],
       [ 3.15660260e+00, -6.52329665e-01],
       [-1.38439716e+00,  6.87031574e-01],
       [-2.72500651e+00, -1.50578390e+00],
       [ 2.39627032e+00, -7.43509614e-01],
       [-2.03660409e+00, -1.65406399e+00],
       [ 2.53455268e-02,  2.19132518e+00],
       [ 1.46233278e+00,  1.06568639e+00],
       [ 8.35508538e-01,  2.11097938e+00],
       [-7.29346316e-01,  1.32960959e+00],
       [-1.79077244e+00,  2.96363426e-01],
       [-3.12416224e+00, -1.94659598e-01],
       [-2.09980872e+00, -1.07101401e+00],
       [-8.76644205e-01,  2.18584169e+00],
       [ 3.52641939e+00, -1.18546709e+00],
       [ 4.79133113e-01,  2.03334080e+00],
       [ 1.44842333e+00,  1.93539177e+00],
       [ 2.21927427e+00, -1.37873951e+00],
       [ 2.42023302e+00, -5.63879500e-01],
       [ 3.69835717e-01,  1.80807276e+00],
       [-2.67222651e+00, -1.08431555e-01],
       [ 3.05863948e+00, -3.03617688e+00],
       [-1.75602185e+00, -7.63362137e-01],
       [ 7.60287931e-01,  9.40333283e-01],
       [-3.51948626e+00, -1.31289347e+00],
       [-2.72592237e+00, -1.24741233e+00],
       [-1.40683739e+00, -6.89118430e-01],
       [ 3.28205726e+00, -2.14011598e-03],
       [-2.63168489e+00, -1.25007176e+00],
       [ 7.09705398e-01,  2.15178745e+00],
       [-1.73277249e+00,  9.70274480e-01],
       [ 8.18614144e-01, -3.67383172e+00],
       [ 1.14664443e+00,  1.63506371e+00],
       [-3.51589433e+00, -1.62337177e+00],
       [ 2.30930778e+00, -2.52199584e+00],
       [-2.06096375e+00, -1.29724906e+00],
       [-2.16946695e+00,  1.39336494e+00],
       [-2.61026189e+00, -1.04965897e+00],
```

```
[-2.28190542e+00, -6.90087471e-01],
[ 2.78299710e+00, -1.83747001e+00],
[ 2.93674084e+00, -4.71370911e-01],
[-2.72779185e+00, -8.10363613e-01],
[ 1.76034445e-01,  1.98386044e+00],
[ 3.72931157e+00, -1.13889219e+00],
[ 2.93383608e+00, -7.13311461e-01],
[ 2.77126867e+00, -6.56755509e-01],
[ 2.75773098e+00, -5.13910651e-01],
[-8.68148076e-01,  2.48449890e+00],
[ 2.65777330e+00, -8.33495488e-01],
[-2.13778709e+00, -1.04192529e+00],
[-2.20959301e+00, -1.25770185e+00],
[ 5.03422216e-01,  2.29105493e+00],
[-3.87347187e-02,  2.31477406e+00],
[-6.24989203e-01,  3.31710204e+00],
[ 2.93647784e+00, -1.80931696e+00],
[-1.10844952e+00, -1.89542643e-01],
[-3.37992920e+00, -1.40349206e+00],
[ 2.20076222e+00, -2.76333441e+00],
[ 2.30344908e+00, -5.04367820e-01],
[ 3.25199859e+00, -1.59059006e+00],
[ 1.69682108e-01,  1.06367659e+00],
[-1.45761817e+00, -7.37363140e-01],
[ 1.58453255e+00,  1.69523160e+00],
[ 5.21835944e-01,  1.82247711e+00],
[ 3.93574081e+00, -4.78354606e-01],
[ 3.69101643e-01,  2.69967190e+00],
[ 3.12333560e+00, -6.48609958e-01],
[-3.60658600e+00, -2.53844883e+00],
[ 1.04561683e+00,  5.38598081e-01],
[-1.61954629e+00, -1.17617419e-01],
[-4.39695627e-01, -4.54939493e-01],
[ 2.63265820e+00, -9.34343705e-01],
[-2.88953774e+00, -7.46236460e-01],
[-1.94979919e+00,  1.44784395e+00],
[ 1.79941935e+00,  1.51864781e+00],
[-2.20317548e+00, -2.43316147e-01],
[-3.15406193e+00, -7.42899168e-01],
[-1.63038482e+00, -2.03364838e-01],
[-1.36959502e+00,  2.09334766e+00],
[-8.06966461e-01,  1.44290986e+00],
[ 1.31264842e+00,  2.85376330e+00],
[-3.17468394e+00, -1.73611071e+00],
[ 2.52150406e+00, -1.99932056e+00],
[ 2.47106252e+00, -2.21878807e+00],
[ 1.62741785e+00,  7.09287328e-01],
[ 1.66501931e+00,  1.17978050e+00],
[ 1.66144912e+00,  1.17326273e+00],
[-2.30691345e+00, -1.91642518e+00],
[ 8.88076937e-01,  5.41153357e-02],
[-4.33957285e-01,  3.79568990e+00],
[-2.06385782e+00,  2.23735281e-02],
[-9.00729612e-01,  7.12082215e-01],
```

```
       [-1.14645976e+00, -8.99119984e-01],
       [-2.76796766e+00, -1.44360282e+00],
       [-1.24049927e+00, -9.32164236e-01],
       [-4.35627225e-01,  9.33121513e-01],
       [-1.14344572e+00, -2.75539335e-02],
       [-1.40225083e+00,  1.33379248e+00],
       [-2.81999469e+00, -1.75152348e-01],
       [-1.61092018e+00,  1.45082255e+00],
       [ 4.89405844e-01,  4.35213588e-01],
       [ 2.60305963e+00, -3.01128944e+00],
       [ 4.37555365e+00, -9.97564557e-01],
       [ 4.54946567e-01,  3.44750807e-01],
       [ 6.63104666e-01,  2.04821361e+00],
       [-6.40780611e-01,  2.27319017e+00],
       [-1.93858807e+00, -2.06224467e-01],
       [ 6.20870162e-01,  2.50688596e+00],
       [ 5.09696563e-01,  2.26978153e+00],
       [-3.40689729e+00, -1.17262793e+00],
       [-2.92797248e-01,  1.98243448e+00],
       [ 1.23010630e-01,  1.85975483e+00],
       [ 4.85445213e-01,  1.82532445e+00],
       [-1.81625231e+00,  1.26506085e+00],
       [ 2.62885951e+00,  9.81236766e-01],
       [ 2.78587519e+00, -7.86935558e-01],
       [ 2.34077720e+00, -6.04948931e-01],
       [ 2.16376934e+00, -5.52830261e-01],
       [-2.75356269e+00, -1.78376482e+00],
       [-2.50652630e+00,  7.19401516e-02],
       [-1.66677328e-01,  2.04664925e+00],
       [-3.84351279e+00, -2.75980408e+00],
       [ 1.38330888e+00, -2.93748220e-01],
       [ 2.46919839e+00,  1.95370955e-01],
       [ 3.55229124e+00, -2.14221861e+00],
       [ 1.03705712e+00,  2.29899043e+00]])
```

In [47]:
```
from sklearn.linear_model import LogisticRegression
pc_model = LogisticRegression()
pc_model.fit(tr_comp,y_train)
```

Out[47]:
```
▾ LogisticRegression

LogisticRegression()
```

In [49]:
```
y_predict=pc_model.predict(ts_comp)
y_predict
```

Out[49]:
```
array([3, 3, 2, 1, 2, 3, 1, 3, 2, 3, 3, 3, 2, 2, 3, 2, 1, 1, 2, 3, 1, 1,
       2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 3, 1, 3])
```

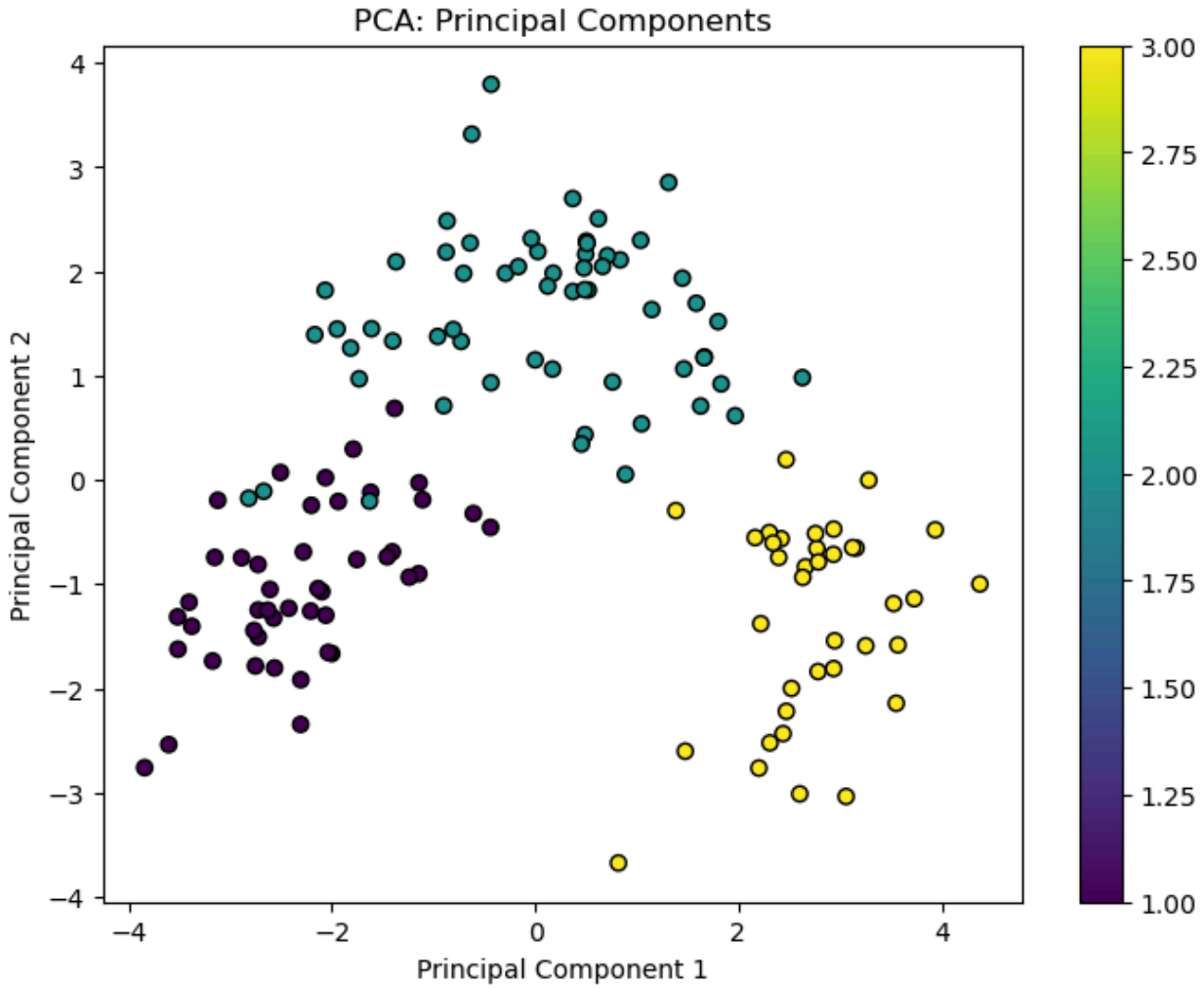
In [51]:
```
y_actual=y_test
print(classification_report(y_actual,y_predict))
```

```
              precision    recall  f1-score   support

           1       1.00      0.86      0.92        14
           2       0.85      1.00      0.92        11
           3       1.00      1.00      1.00        11

    accuracy                           0.94        36
   macro avg       0.95      0.95      0.95        36
weighted avg       0.95      0.94      0.94        36
```

In [53]:
```
principal_components = pca.components_
print("Principal Components:")
print(principal_components)
```

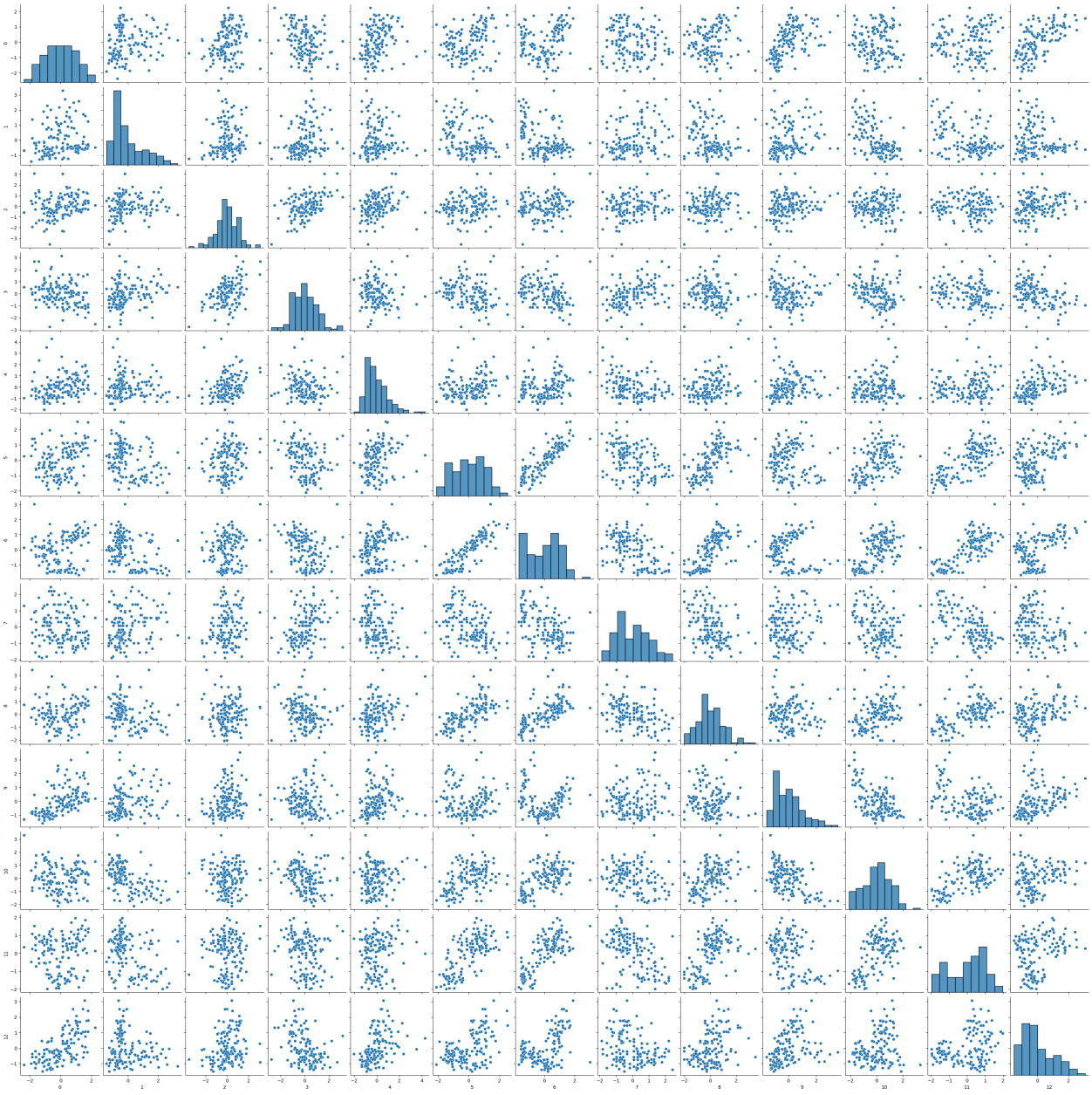
```
Principal Components:
[[-0.14067905  0.31993109  0.09290431  0.35367843 -0.02473001 -0.34130895
  -0.36115592  0.31689445 -0.20114095  0.1522117  -0.27887711 -0.39528538
  -0.3138881 ]
 [-0.53680044 -0.15301763 -0.20966357  0.02815284 -0.2316119  -0.04787013
  -0.02141232  0.06707401 -0.03041416 -0.55218445  0.26407493  0.13421415
  -0.43497675]]
```

In [55]:
```
import matplotlib.pyplot as plt
# Create a scatter plot to visualize the principal components
plt.figure(figsize=(8, 6))
plt.scatter(tr_comp[:, 0], tr_comp[:, 1], c=y_train, cmap='viridis', edgecolor
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA: Principal Components')
plt.colorbar()
plt.show()
```



In [57]:
```
sns.pairplot(X_train)
```

Out[57]:
```
<seaborn.axisgrid.PairGrid at 0x7fde67586d00>
```



In [ ]: