In [48]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# Data processing, modeling, and model evaluation
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, classification_report, Confusio
```

In [49]:
```
dataset = pd.read_csv("ep.csv")
df = pd.DataFrame(dataset)
df.head()
```

Out[49]:

	Unnamed	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X170	X171	X172	X173
0	X21.V1.791	135	190	229	223	192	125	55	-9	-33	...	-17	-15	-31	-77
1	X15.V1.924	386	382	356	331	320	315	307	272	244	...	164	150	146	152
2	X8.V1.1	-32	-39	-47	-37	-32	-36	-57	-73	-85	...	57	64	48	19
3	X16.V1.60	-105	-101	-96	-92	-89	-95	-102	-100	-87	...	-82	-81	-80	-77
4	X20.V1.54	-9	-65	-98	-102	-78	-48	-16	0	-21	...	4	2	-12	-32

5 rows × 180 columns

In [50]:
```
df.pop('Unnamed')
```

Out[50]:
```
0         X21.V1.791
1         X15.V1.924
2            X8.V1.1
3          X16.V1.60
4          X20.V1.54
              ...
11495    X22.V1.114
11496    X19.V1.354
11497      X8.V1.28
11498    X10.V1.932
11499    X16.V1.210
Name: Unnamed, Length: 11500, dtype: object
```

In [51]:
```
df.head()
```

Out[51]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X170	X171	X172	X173	X174
0	135	190	229	223	192	125	55	-9	-33	-38	...	-17	-15	-31	-77	-103
1	386	382	356	331	320	315	307	272	244	232	...	164	150	146	152	157
2	-32	-39	-47	-37	-32	-36	-57	-73	-85	-94	...	57	64	48	19	-12
3	-105	-101	-96	-92	-89	-95	-102	-100	-87	-79	...	-82	-81	-80	-77	-85
4	-9	-65	-98	-102	-78	-48	-16	0	-21	-59	...	4	2	-12	-32	-41

5 rows × 179 columns

In [52]:
```
df['y'] = [1 if x == 1 else 0 for x in df['y']]
```

In [53]:
```
df.head()
```

Out[53]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X170	X171	X172	X173	X174
0	135	190	229	223	192	125	55	-9	-33	-38	...	-17	-15	-31	-77	-103
1	386	382	356	331	320	315	307	272	244	232	...	164	150	146	152	157
2	-32	-39	-47	-37	-32	-36	-57	-73	-85	-94	...	57	64	48	19	-12
3	-105	-101	-96	-92	-89	-95	-102	-100	-87	-79	...	-82	-81	-80	-77	-85
4	-9	-65	-98	-102	-78	-48	-16	0	-21	-59	...	4	2	-12	-32	-41

5 rows × 179 columns

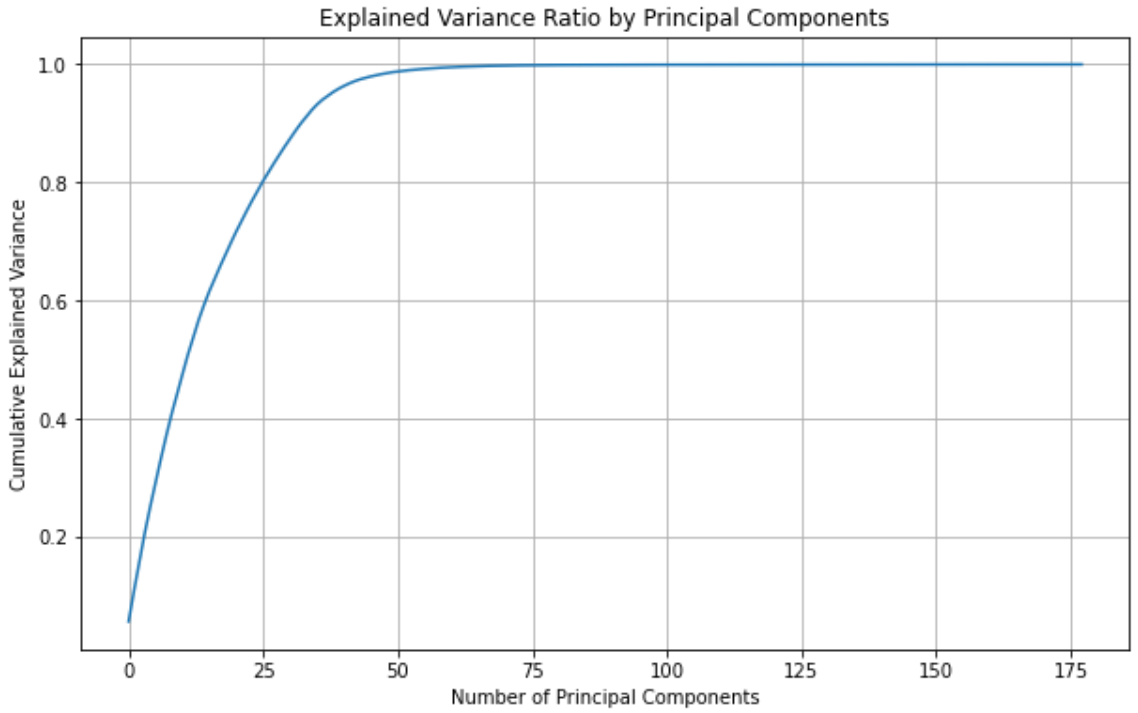
In [54]:
```
sum(df.isnull().sum())
```

Out[54]:
```
0
```

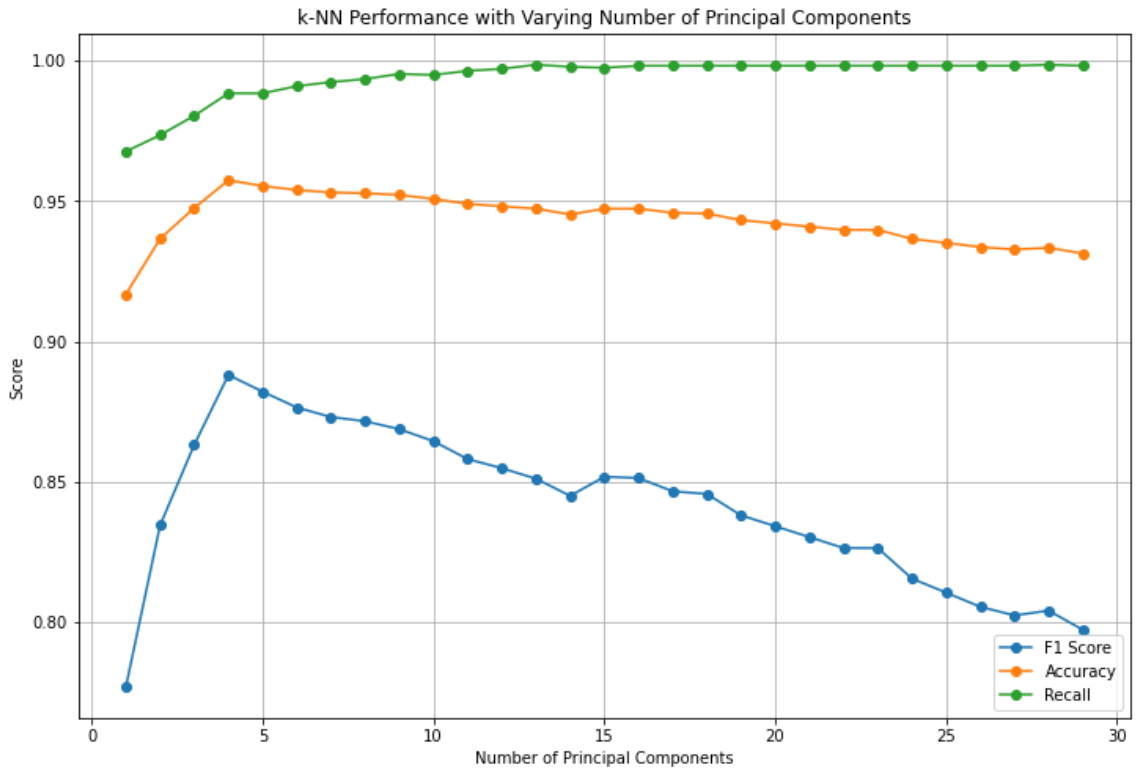
In [55]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [61]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [57]:
```
pca = PCA(n_components=178)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(10, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance Ratio by Principal Components')
plt.grid(True)
plt.show()
```

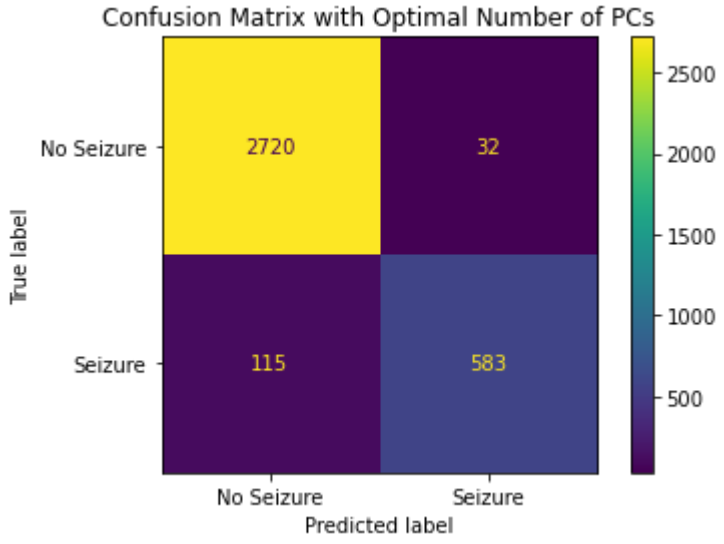


In [58]:
```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_si

results = {'num_pcs': [], 'f1_score': [], 'accuracy': [], 'recall': |

for n_pcs in range(1, 20):
    X_train_reduced = X_train[:, :n_pcs]
    X_test_reduced = X_test[:, :n_pcs]

    knn = KNeighborsClassifier()
    knn.fit(X_train_reduced, y_train)

    y_pred = knn.predict(X_test_reduced)
    f1 = f1_score(y_test, y_pred)
    accuracy = np.mean(y_pred == y_test)
    recall = classification_report(y_test, y_pred, output_dict=True)|

    results['num_pcs'].append(n_pcs)
    results['f1_score'].append(f1)
    results['accuracy'].append(accuracy)
    results['recall'].append(recall)


results_df = pd.DataFrame(results)

plt.figure(figsize=(12, 8))
plt.plot(results_df['num_pcs'], results_df['f1_score'], marker='o', l
plt.plot(results_df['num_pcs'], results_df['accuracy'], marker='o', l
plt.plot(results_df['num_pcs'], results_df['recall'], marker='o', lab
plt.xlabel('Number of Principal Components')
plt.ylabel('Score')
plt.title('k-NN Performance with Varying Number of Principal Componer
plt.legend()
plt.grid(True)
plt.show()
```



In [64]:
```
print(optimal_n_pcs)

X_train_opt = X_train[:, :optimal_n_pcs]
X_test_opt = X_test[:, :optimal_n_pcs]

knn_opt = KNeighborsClassifier()
knn_opt.fit(X_train_opt, y_train)

y_pred_opt = knn_opt.predict(X_test_opt)
accuracy_opt = np.mean(y_pred_opt == y_test)
f1_opt = f1_score(y_test, y_pred_opt)
report_opt = classification_report(y_test, y_pred_opt)

cm = confusion_matrix(y_test, y_pred_opt)

cm_display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labe

plt.figure(figsize=(8, 8))
cm_display.plot(values_format='d')
plt.title('Confusion Matrix with Optimal Number of PCs')
plt.show()

print(f"Optimal number of PCs: {optimal_n_pcs}")
print(f"Accuracy: {accuracy_opt:.4f}")
print(f"F1 Score: {f1_opt:.4f}")
print("Classification Report:")
print(report_opt)
```

```
4

<Figure size 576x576 with 0 Axes>
```



```
Optimal number of PCs: 4
Accuracy: 0.9574
F1 Score: 0.8880
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      2752
           1       0.95      0.84      0.89       698

    accuracy                           0.96      3450
   macro avg       0.95      0.91      0.93      3450
weighted avg       0.96      0.96      0.96      3450
```

In [ ]: