In [1]:
```
#KNN CLASSIFIER

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("DATA/gene_expression.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Gene One	Gene Two	Cancer Present
0	4.3	3.9	1
1	2.5	6.3	0
2	5.7	3.9	1
3	6.1	6.2	0
4	7.4	3.4	1

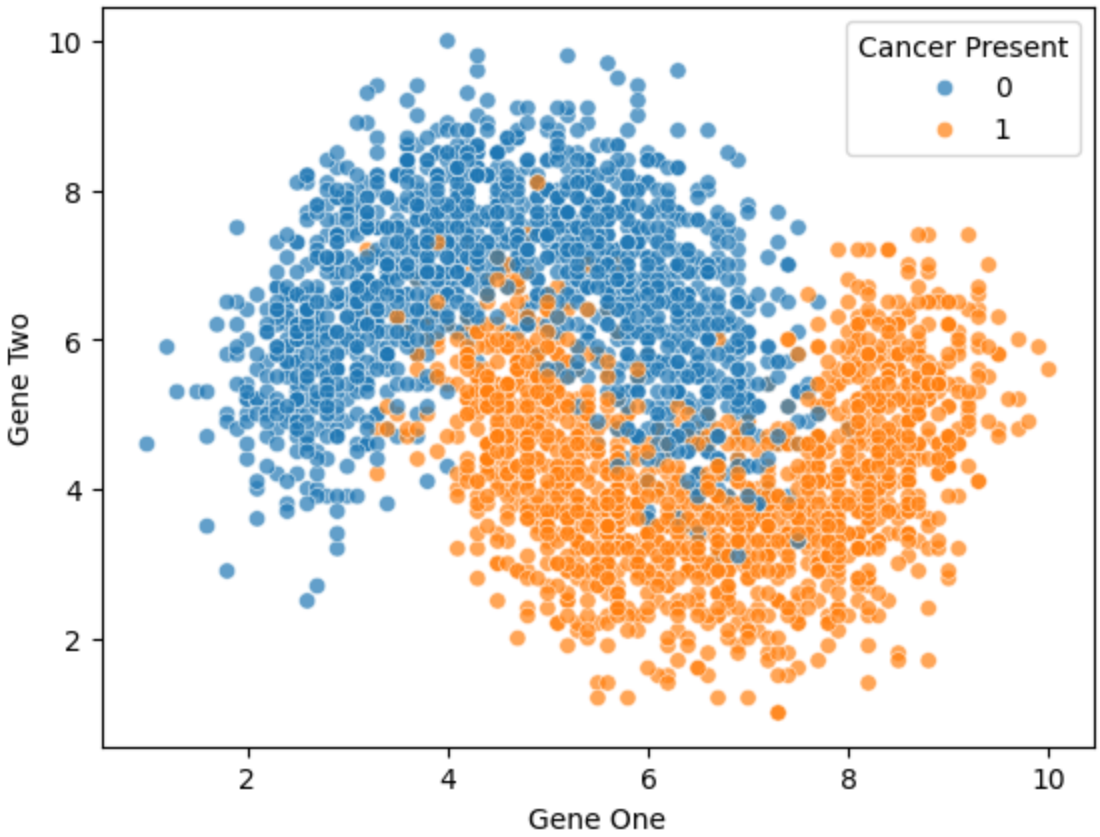
In [5]:
```
df.describe()
```

Out[5]:

	Gene One	Gene Two	Cancer Present
count	3000.000000	3000.000000	3000.000000
mean	5.600133	5.410467	0.500000
std	1.828388	1.729081	0.500083
min	1.000000	1.000000	0.000000
25%	4.300000	4.000000	0.000000
50%	5.600000	5.400000	0.500000
75%	6.900000	6.700000	1.000000
max	10.000000	10.000000	1.000000

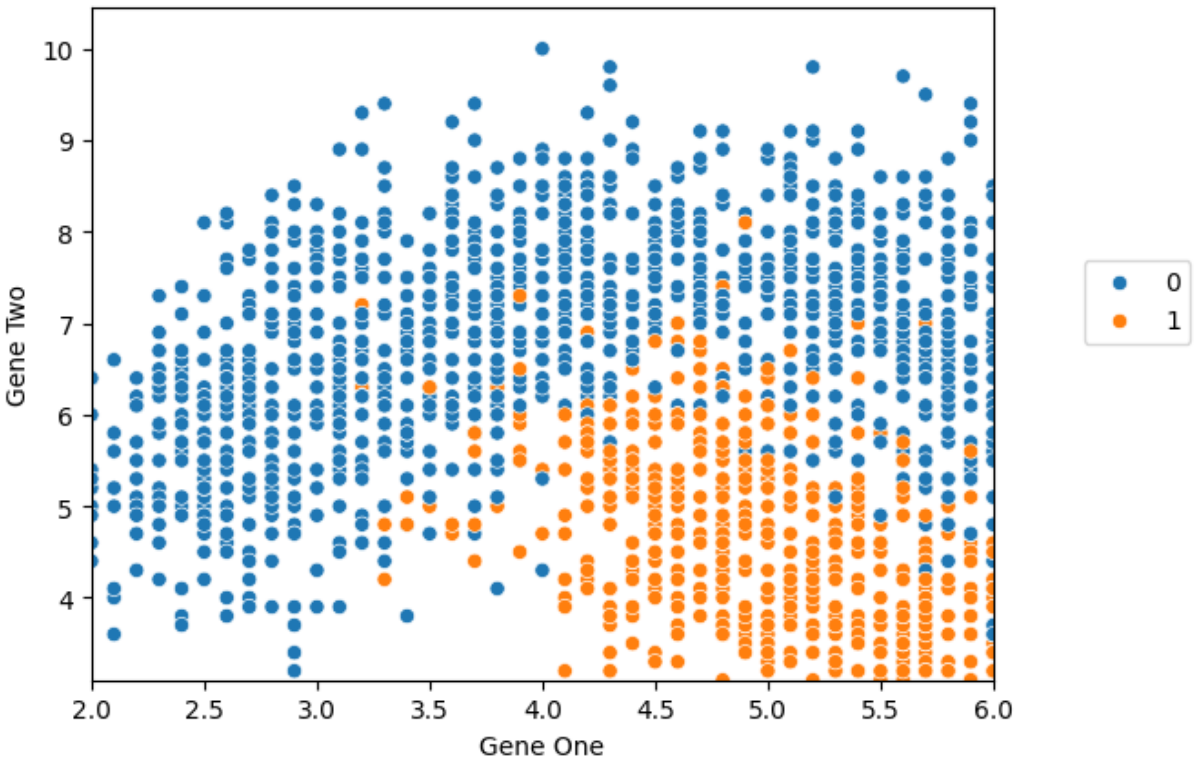
In [7]:
```
sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data=df,alpha=0.7)
```

Out[7]:
```
<Axes: xlabel='Gene One', ylabel='Gene Two'>
```



In [9]:
```
sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data=df)
plt.xlim(2,6)
plt.ylim(3.10)
plt.legend(loc=(1.1,0.5))
```

Out[9]: <matplotlib.legend.Legend at 0x22ca303be30>



In [10]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [12]:
```
X = df.drop('Cancer Present', axis=1)
```

In [13]:
```
y=df['Cancer Present']
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state
```

In [15]:
```
scaler=StandardScaler()
```

In [16]:
```
scaled_X_train=scaler.fit_transform(X_train)
```

In [17]:
```
scaled_X_test=scaler.fit_transform(X_test)
```

In [18]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [53]:
```
knn_model =  KNeighborsClassifier(n_neighbors=5)
knn_model.fit(scaled_X_train, y_train)
```

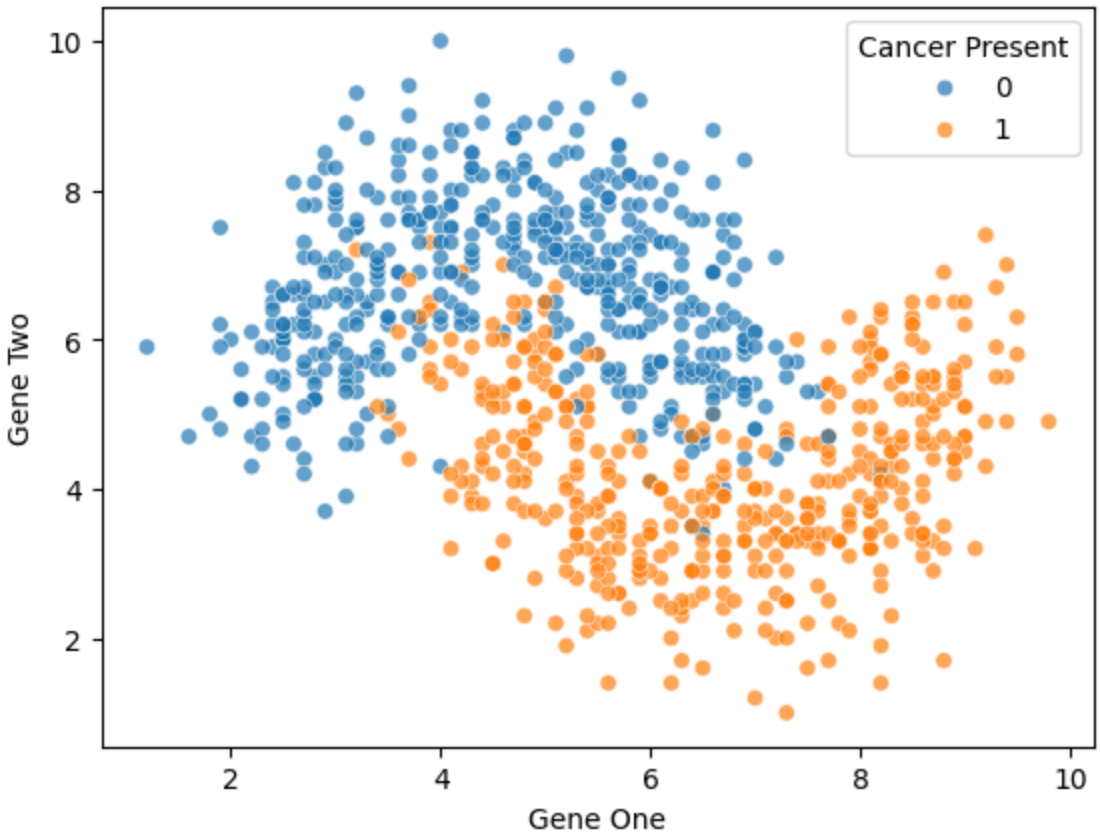
Out[53]:
```
▾  KNeighborsClassifier i ?

KNeighborsClassifier()
```

In [54]:
```
full_test= pd.concat([X_test,y_test],axis=1)
len(full_test)
```

Out[54]:
```
900
```

In [55]:
```
sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data=full_test,alpha
```

Out[55]:
```
<Axes: xlabel='Gene One', ylabel='Gene Two'>
```



In [56]:
```
y_pred=knn_model.predict(scaled_X_test)
```

In [57]:
```
y_pred
```

Out[57]:
```
array([1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0])
```

In [58]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [59]:
```
accuracy_score(y_test,y_pred)
```

Out[59]:
```
0.9344444444444444
```

In [60]:
```
confusion_matrix(y_test,y_pred)
```

Out[60]:
```
array([[437,  33],
       [ 26, 404]])
```

In [61]:
```
print(classification_report(y_test,y_pred))
```

```
                precision    recall  f1-score   support

           0       0.94      0.93      0.94       470
           1       0.92      0.94      0.93       430

    accuracy                           0.93       900
   macro avg       0.93      0.93      0.93       900
weighted avg       0.93      0.93      0.93       900
```

In [31]:
```
test_error_rates=[]
for k in range(1,30):
    knn_model =KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(scaled_X_train,y_train)
    y_pred_test = knn_model.predict(scaled_X_test)

    test_error=1-accuracy_score(y_test,y_pred_test)
    test_error_rates.append(test_error)
```

In [32]:
```
test_error_rates
```

Out[32]:
```
[0.08666666666666667,
 0.08333333333333337,
 0.07111111111111112,
 0.06999999999999995,
 0.06555555555555559,
 0.06666666666666665,
 0.06555555555555559,
 0.06666666666666665,
 0.06444444444444442,
 0.06777777777777783,
 0.07111111111111112,
 0.06555555555555559,
 0.06999999999999995,
 0.06555555555555559,
 0.06777777777777783,
 0.06666666666666665,
 0.06888888888888889,
 0.06555555555555559,
 0.06555555555555559,
 0.06333333333333335,
 0.06555555555555559,
 0.061111111111111116,
 0.06222222222222218,
 0.06000000000000005,
 0.061111111111111116,
 0.061111111111111116,
 0.06222222222222218,
 0.061111111111111116,
 0.061111111111111116]
```

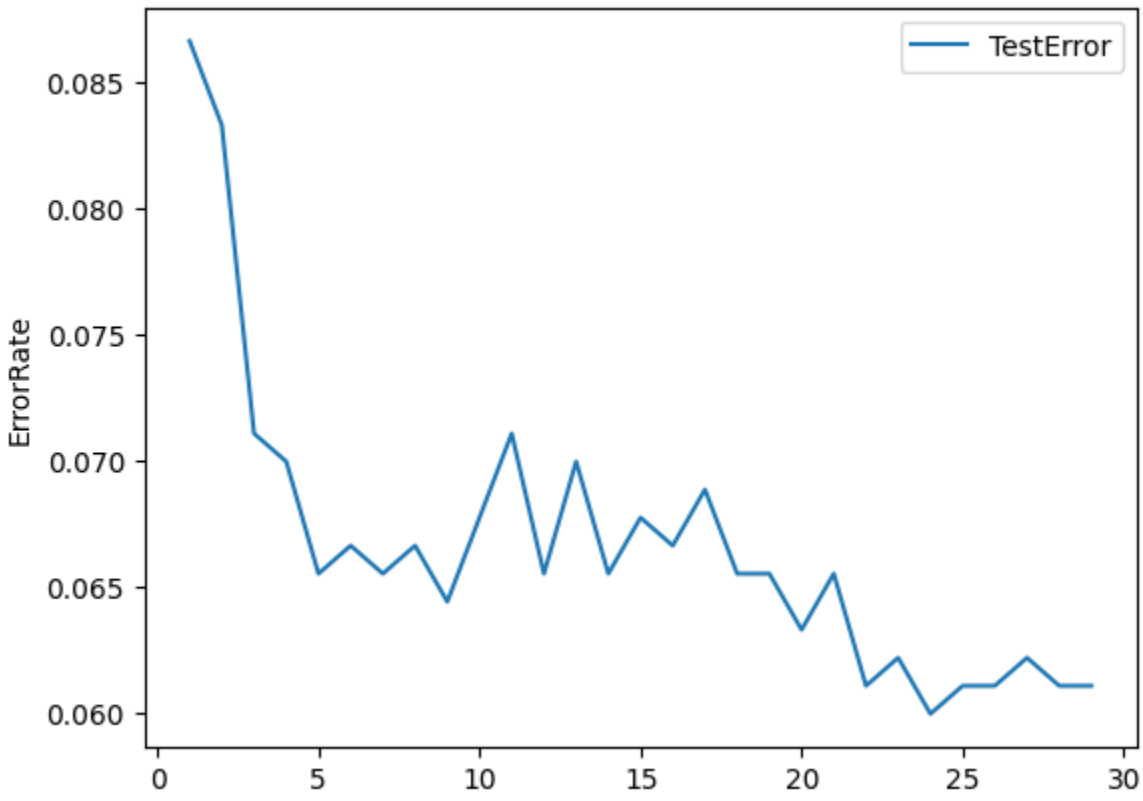
In [33]:
```
plt.figure(figsize=(10,6),dpi=200)
```

Out[33]:
```
<Figure size 2000x1200 with 0 Axes>
```

<Figure size 2000x1200 with 0 Axes>

In [34]:
```
plt.plot(range(1,30),test_error_rates,label='TestError')
plt.legend()
plt.ylabel('ErrorRate')
```

Out[34]: Text(0, 0.5, 'ErrorRate')



In [ ]: