Importing Basic Libraries

In [1]:

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

Getting the dataframe

In [2]:

```
df = pd.read_csv('gene_expression.csv')
```

In [3]:

```
df.head()
```

Out[3]:

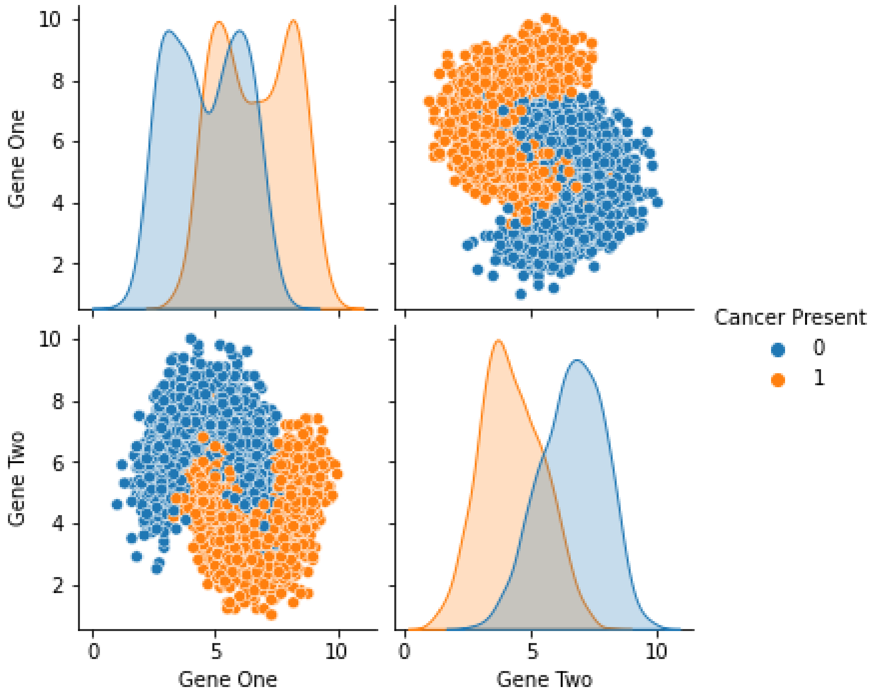
	Gene One	Gene Two	Cancer Present
0	4.3	3.9	1
1	2.5	6.3	0
2	5.7	3.9	1
3	6.1	6.2	0
4	7.4	3.4	1

In [4]:

```
sns.pairplot(df, hue = 'Cancer Present')
```

Out[4]:

<seaborn.axisgrid.PairGrid at 0x13acb6c3e88>

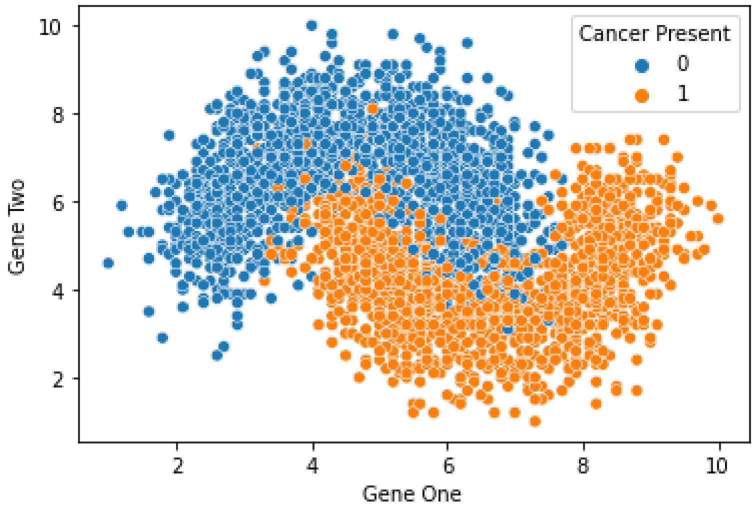


In [5]:

```
sns.scatterplot(x=df["Gene One"], y= df["Gene Two"], hue = df["Cancer Present"])
```

Out[5]:

```
<AxesSubplot:xlabel='Gene One', ylabel='Gene Two'>
```



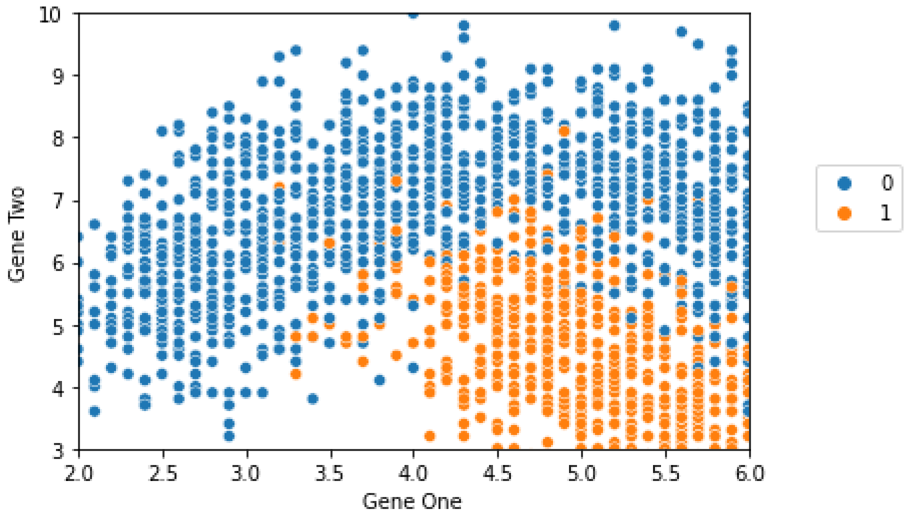
Lets zoom in a little

In [6]:

```
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[6]:

```
<matplotlib.legend.Legend at 0x13acc178288>
```



Now, Lets Split the data into train and test

In [7]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [8]:

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

In [9]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=42)
```

In [10]:

```
scaler = StandardScaler()
scaler.fit(X_train)
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

Train the model

In [11]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [12]:

```
model = KNeighborsClassifier(n_neighbors=15)
model.fit(scaled_X_train,y_train)
```

Out[12]:

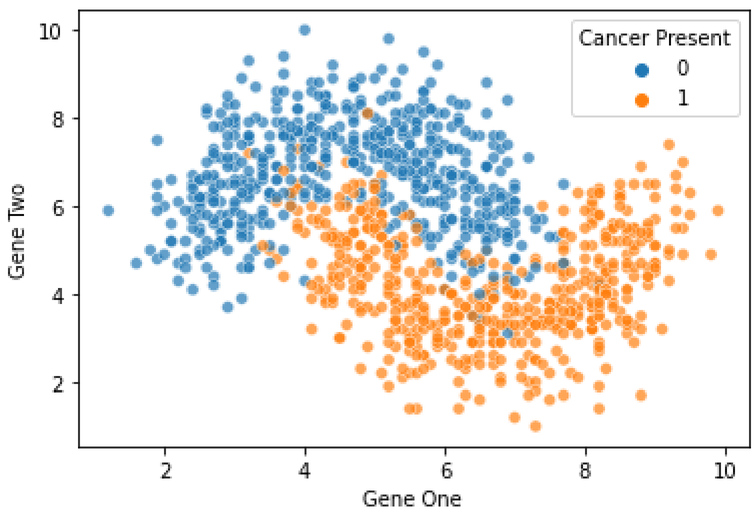
```
KNeighborsClassifier(n_neighbors=15)
```

In [13]:

```
sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data= pd.concat([X_test,y_te
```

Out[13]:

```
<AxesSubplot:xlabel='Gene One', ylabel='Gene Two'>
```



Finally, evaluate the model

In [14]:

```
prediction = model.predict(scaled_X_test)
```

In [15]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [16]:

```
print("Accuracy Score: ", accuracy_score(y_test, prediction))
```

```
Accuracy Score:  0.9457142857142857
```

In [17]:

```
confusion_matrix(y_test,prediction)
```

Out[17]:

```
array([[516,  24],
       [ 33, 477]], dtype=int64)
```

In [18]:

```
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       540
           1       0.95      0.94      0.94       510

    accuracy                           0.95      1050
   macro avg       0.95      0.95      0.95      1050
weighted avg       0.95      0.95      0.95      1050
```

Lets try to optimize the model for best value of K

In [19]:

```
error_rate = []

for i in range(1,50):
    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(scaled_X_train,y_train)
    prediction = model.predict(scaled_X_test)
    error_rate.append(np.mean(prediction != y_test))
```

In [20]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,50), error_rate, color='blue', linestyle='dashed', marker='o', markerfacec
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[20]:

```
Text(0, 0.5, 'Error Rate')
```

We see that the error is least when k = 20

In [21]:

```
model = KNeighborsClassifier(n_neighbors=20)
model.fit(scaled_X_train,y_train)
prediction = model.predict(scaled_X_test)
```

In [22]:

```
print("Accuracy Score: ", accuracy_score(y_test, prediction))
```

```
Accuracy Score:  0.9476190476190476
```

In [23]:

```
confusion_matrix(y_test,prediction)
```

Out[23]:

```
array([[518,  22],
       [ 33, 477]], dtype=int64)
```

In [24]:

```
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       540
           1       0.96      0.94      0.95       510

    accuracy                           0.95      1050
   macro avg       0.95      0.95      0.95      1050
weighted avg       0.95      0.95      0.95      1050
```

We also get the best accuracy for k = 20, hence our task is complete