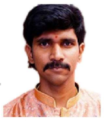


In [1]:

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

import warnings
warnings.simplefilter("ignore")
```

In [2]:

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

Out[2]:

	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

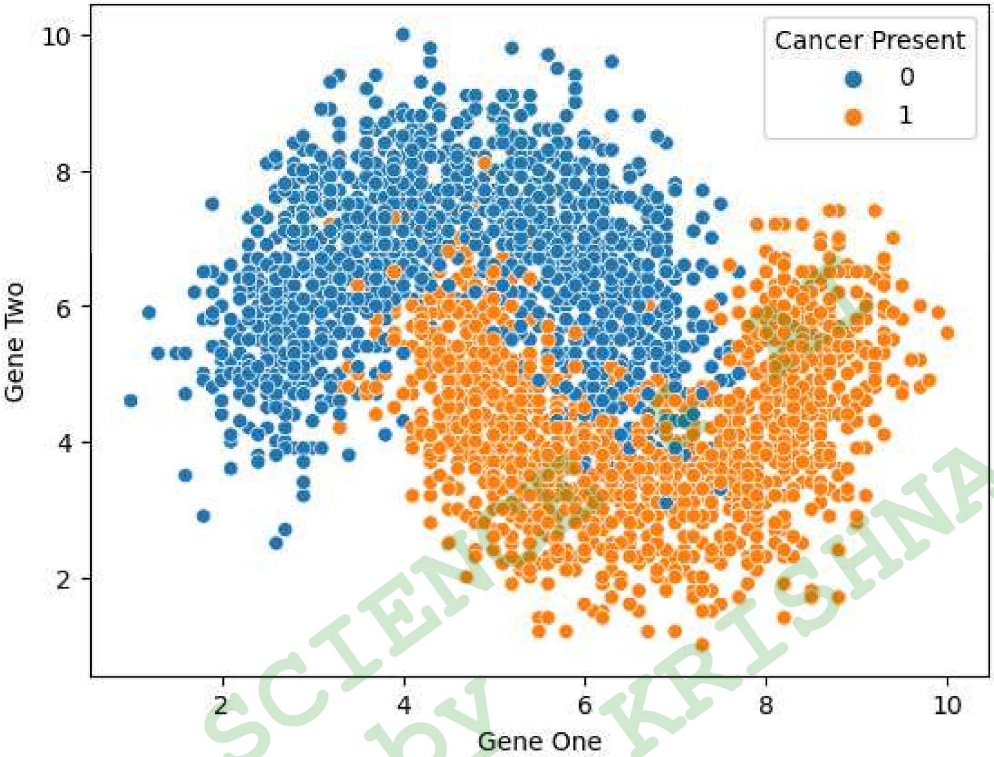
In [3]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Gene One        3000 non-null   float64
 1   Gene Two        3000 non-null   float64
 2   Cancer Present  3000 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 70.4 KB
```

In [4]:

```
sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data=df
plt.show()
```



In [5]:

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

Out[5]:

```
Gene One          0
Gene Two          0
Cancer Present    0
dtype: int64
```

X & y

In [6]:

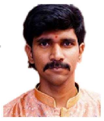
```
X = df.drop('Cancer Present',axis=1)
y = df['Cancer Present']
```

Train|Test Split

In [7]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=9)
```

Scaling Data



In [8]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Hyperparameter Tuning for KNN Classifier

In [9]:

```
from sklearn.neighbors import KNeighborsClassifier
estimator = KNeighborsClassifier()
param_grid = {'n_neighbors': list(range(1,100))}

from sklearn.model_selection import GridSearchCV
cv_classifier = GridSearchCV(estimator, param_grid,
                             cv=5,scoring='accuracy')

cv_classifier.fit(X_train,y_train)

cv_classifier.best_params_
```

Out[9]:

```
{'n_neighbors': 19}
```

KNN model with best hyper parameters

In [10]:

```
#Modelling
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=19)
knn.fit(X_train,y_train)

# Prediction
y_pred_test = knn.predict(X_test)
y_pred_train = knn.predict(X_train)

#Evaluation
from sklearn.metrics import accuracy_score
print("Train accuracy:",accuracy_score(y_train,y_pred_train))
print("Test accuracy:",accuracy_score(y_test,y_pred_test))

from sklearn.model_selection import cross_val_score
print("Cross Validation Score:",cross_val_score(knn,X,y,cv=5).mean())
```

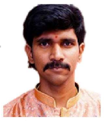
```
Train accuracy: 0.9408333333333333
Test accuracy: 0.9333333333333333
Cross Validation Score: 0.9313333333333335
```

In [11]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred_test)
```

Out[11]:

```
array([[264,  19],
       [ 21, 296]], dtype=int64)
```



In [12]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       283
           1       0.94      0.93      0.94       317

    accuracy                           0.93       600
   macro avg       0.93      0.93      0.93       600
weighted avg       0.93      0.93      0.93       600
```

Prediction on New data

New Data

In [13]:

```
df = pd.DataFrame({"Gene One":[4.9],"Gene Two":[3.9]})
df
```

Out[13]:

	Gene One	Gene Two
0	4.9	3.9

preprocess the new data

In [14]:

```
df_scaled = scaler.transform(df)
```

use KNN model to predict on new data

In [15]:

```
knn.predict(df_scaled)
```

Out[15]:

```
array([1], dtype=int64)
```