In [4]:

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

In [2]:

```
df = pd.read_csv('scaler.csv')
df.shape
```

Out[2]:

```
(400, 5)
```

In [3]:

```
df.head()
```

Out[3]:

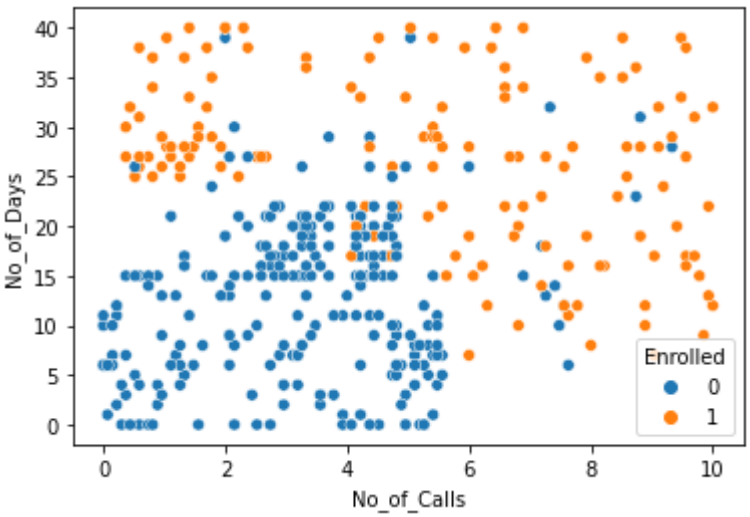
	User_ID	Gender	No_of_Days	No_of_Calls	Enrolled
0	15624510	Male	0	0.30	0
1	15810944	Male	15	0.37	0
2	15668575	Female	6	2.07	0
3	15603246	Female	7	3.11	0
4	15804002	Male	0	4.52	0

In [5]:

```
sns.scatterplot(data=df, x='No_of_Calls', y='No_of_Days', hue='Enrolled')
```

Out[5]:

```
<AxesSubplot:xlabel='No_of_Calls', ylabel='No_of_Days'>
```



In [9]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df[['No_of_Days', 'No_of_Calls']
                                                    df['Enrolled'],
                                                    test_size=0.25,
                                                    random_state=4)
```

In [10]:

```
X_train.shape, X_test.shape
```

Out[10]:

```
((300, 2), (100, 2))
```

In [12]:

```
mu = X_train.mean(axis=0)
sig = X_train.std(axis=0)

X_train = (X_train-mu)/sig
X_test = (X_test-mu)/sig
```

In [ ]:

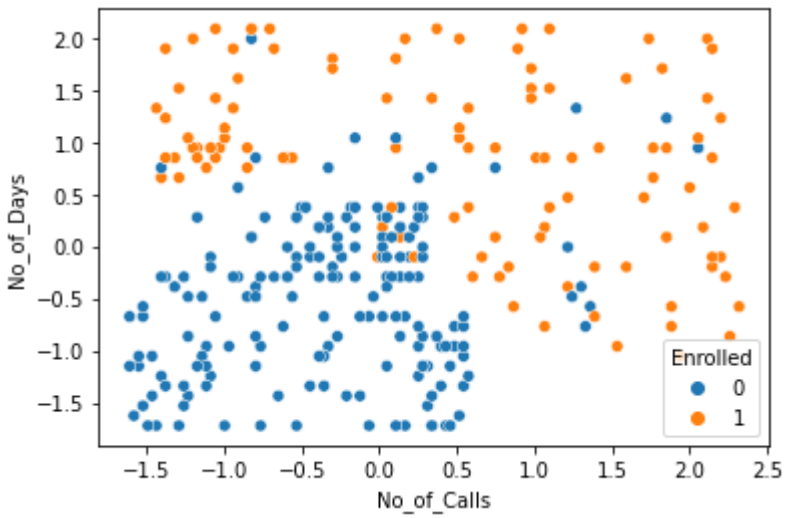
```

```

In [14]:

```
sns.scatterplot(data=X_train, x='No_of_Calls', y='No_of_Days', hue=y_train)
```

Out[14]:

```
<AxesSubplot:xlabel='No_of_Calls', ylabel='No_of_Days'>
```

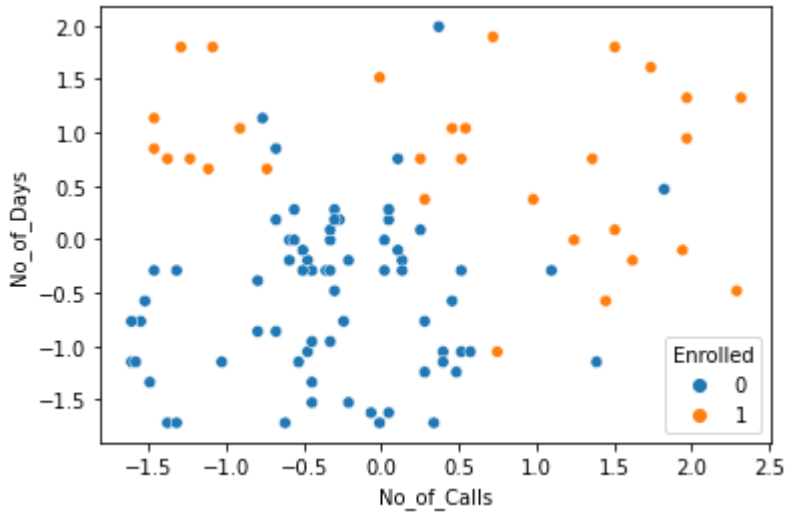


In [15]:

```
sns.scatterplot(data=X_test, x='No_of_Calls', y='No_of_Days', hue=y_test)
```

Out[15]:

```
<AxesSubplot:xlabel='No_of_Calls', ylabel='No_of_Days'>
```



In [ ]:

```

```

In [16]:

```
X_train = X_train.values
X_test = X_test.values

y_train = y_train.values
y_test = y_test.values
```

In [17]:

```
X_train[:5]
```

Out[17]:

```
array([[-1.05194419, -1.46297237],
       [-0.67160884,  0.1668288 ],
       [-1.71753105,  0.45744395],
       [ 1.9907386 , -0.82283359],
       [ 1.42023558,  0.339627  ]])
```

In [18]:

```
y_train[:5]
```

Out[18]:

```
array([0, 0, 0, 0, 1])
```

In [ ]:

```

```

The Algorithm

In [45]:

```
def distance(pA, pB):
    return np.sqrt(np.sum((pA-pB)**2))

def kNNPredict(X_train, y_train, x_query, k=5):

    n = X_train.shape[0]

    all_distances = []

    for i in range(n):
        d = distance(X_train[i], x_query)
        all_distances.append( (d, y_train[i]) )

    distances = sorted(all_distances)
    distances = distances[:k]

    votes  = [lab for dis,lab in distances]

    uniq, counts = np.unique(votes, return_counts=True)
    pred = uniq[np.argmax(counts)]

    return pred
```

In [46]:

```
kNNPredict(X_train, y_train, X_test[68], k=9)
```

Out[46]:

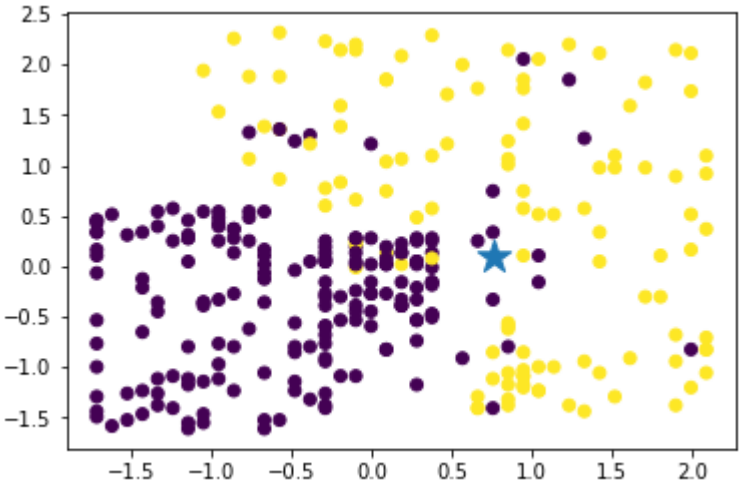
```
0
```

In [47]:

```
plt.scatter(X_train[:, 0], X_train[:, 1], c = y_train)
plt.scatter(X_test[68][0], X_test[68][1], s=300, marker='*')
```

Out[47]:

```
<matplotlib.collections.PathCollection at 0x7ff84265b610>
```



In [ ]:

```
```

In [69]:

```
y_pred = []
for query in X_test:
    p = kNNPredict(X_train, y_train, query, k = 11)
    y_pred.append(p)

y_pred = np.array(y_pred)
```

In [70]:

```
y_test
```

Out[70]:

```
array([0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
1,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1])
```

In [71]:

```
y_pred
```

Out[71]:

```
array([0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
1,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1])
```

In [72]:

```
(y_test==y_pred).mean()
```

Out[72]:

```
0.9
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```