In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

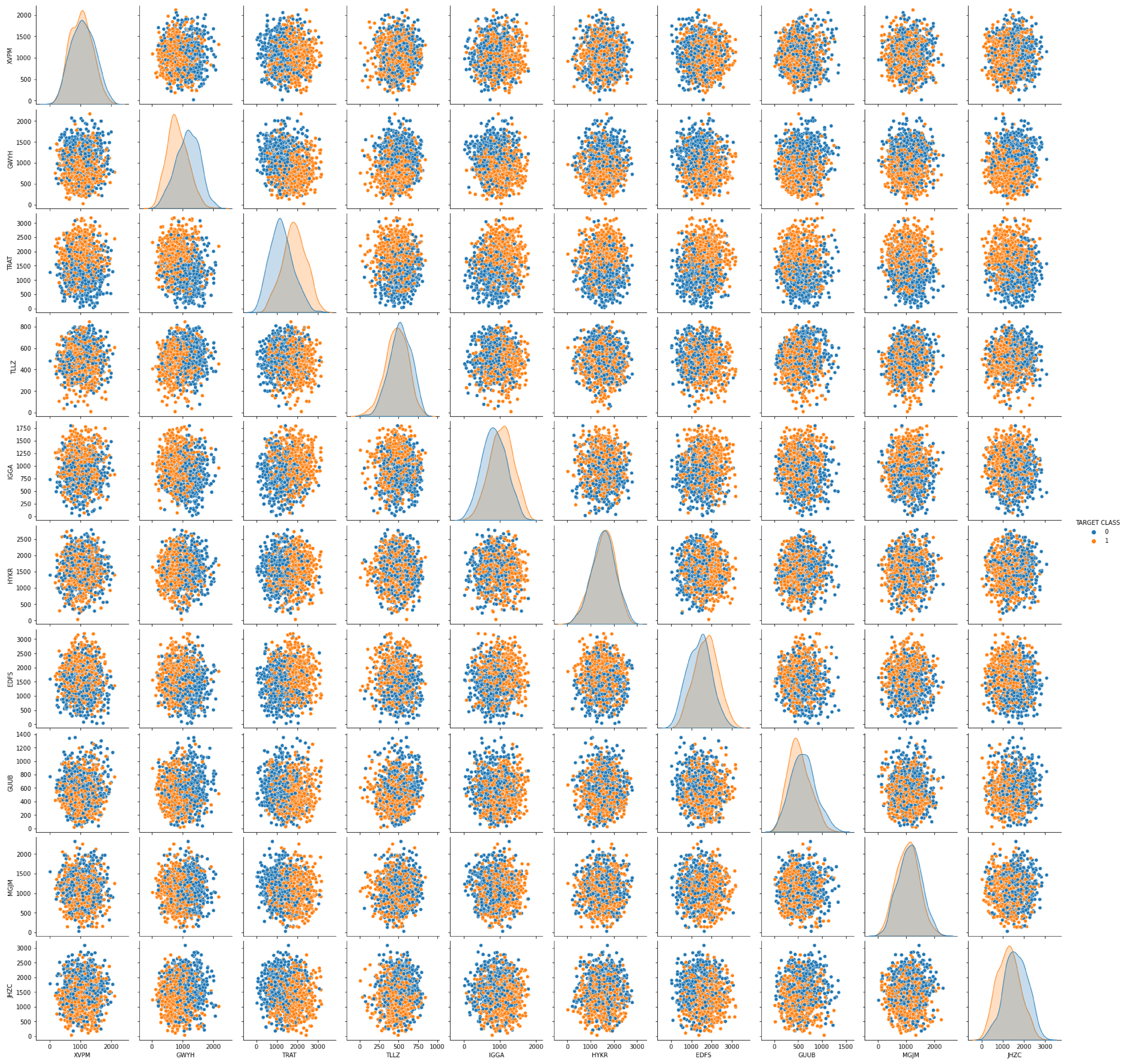
In [5]:
```python
df=pd.read_csv("KNN_Project_Data")
df.head()
```

Out[5]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	JHZC	TARGET CLASS
0	1636.670614	817.988525	2565.995189	358.347163	550.417491	1618.870897	2147.641254	330.727893	1494.878631	845.136088	0
1	1013.402760	577.587332	2644.141273	280.428203	1161.873391	2084.107872	853.404981	447.157619	1193.032521	861.081809	1
2	1300.035501	820.518697	2025.854469	525.562292	922.206261	2552.355407	818.676686	845.491492	1968.367513	1647.186291	1
3	1059.347542	1066.866418	612.000041	480.827789	419.467495	685.666983	852.867810	341.664784	1154.391368	1450.935357	0
4	1018.340526	1313.679056	950.622661	724.742174	843.065903	1370.554164	905.469453	658.118202	539.459350	1899.850792	0

In [7]:
```python
sns.pairplot(data=df,hue="TARGET CLASS")
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x1c040d2a670>



In [9]:
```python
from sklearn.preprocessing import StandardScaler
```

In [10]:
```python
scaler=StandardScaler()
```

In [12]:
```python
scaler.fit(df.drop("TARGET CLASS",axis=1))
```

Out[12]: StandardScaler()

In [14]:
```python
scaled_features=scaler.transform(df.drop("TARGET CLASS",axis=1))
```

In [16]:
```python
df_feat=pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_feat.head()
```

Out[16]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	JHZC
0	1.568522	-0.443435	1.619808	-0.958255	-1.128481	0.138336	0.980493	-0.932794	1.008313	-1.069627
1	-0.112376	-1.056574	1.741918	-1.504220	0.640009	1.081552	-1.182663	-0.461864	0.258321	-1.041546
2	0.660647	-0.436981	0.775793	0.213394	-0.053171	2.030872	-1.240707	1.149298	2.184784	0.342811
3	0.011533	0.191324	-1.433473	-0.100053	-1.507223	-1.753632	-1.183561	-0.888557	0.162310	-0.002793
4	-0.099059	0.820815	-0.904346	1.609015	-0.282065	-0.365099	-1.095644	0.391419	-1.365603	0.787762

In [17]:
```python
from sklearn.model_selection import train_test_split
```

In [18]:
```python
X= df_feat
y= df["TARGET CLASS"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [20]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [21]:
```python
knn=KNeighborsClassifier(n_neighbors=1)
```

In [22]:
```python
knn.fit(X_train,y_train)
```

Out[22]: KNeighborsClassifier(n_neighbors=1)

In [25]:
```python
from sklearn.metrics import classification_report,confusion_matrix
```

In [26]:
```python
pred=knn.predict(X_test)
```

In [29]:
```python
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.73      0.72       163
           1       0.73      0.70      0.71       167

    accuracy                           0.72       330
   macro avg       0.72      0.72      0.72       330
weighted avg       0.72      0.72      0.72       330
```

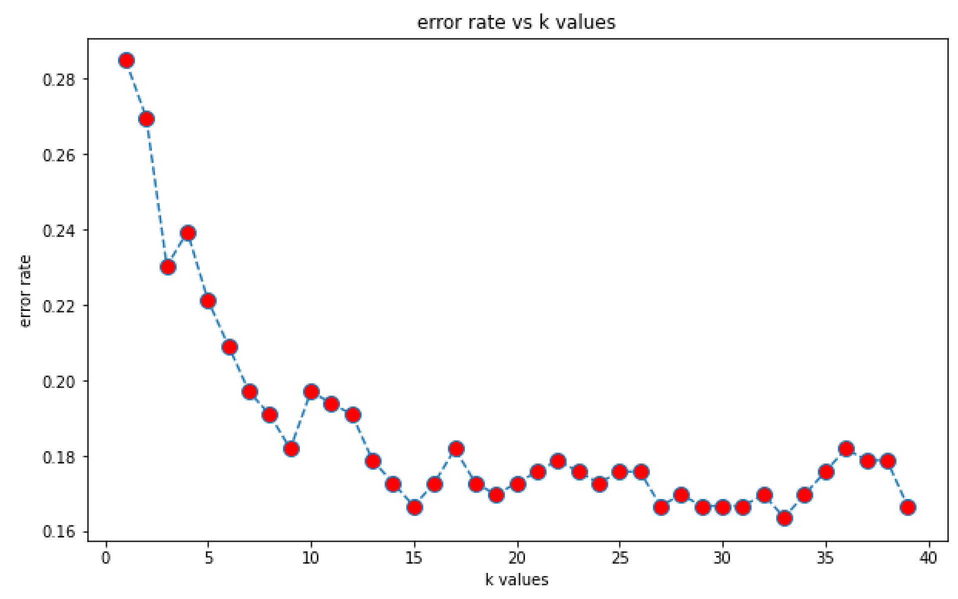
In [30]:
```python
print(confusion_matrix(y_test,pred))
```

```
[[119  44]
 [ 50 117]]
```

In [31]:
```python
error_rate=[]
for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i=knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [35]:
```python
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,ls="--",marker="o",markersize=10,markerfacecolor="red")
plt.xlabel("k values")
plt.ylabel("error rate")
plt.title("error rate vs k values")
```

Out[35]: Text(0.5, 1.0, 'error rate vs k values')



In [36]:
```python
knn = KNeighborsClassifier(n_neighbors=30)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=30')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=30


[[141  22]
 [ 33 134]]


              precision    recall  f1-score   support

           0       0.81      0.87      0.84       163
           1       0.86      0.80      0.83       167

    accuracy                           0.83       330
   macro avg       0.83      0.83      0.83       330
weighted avg       0.83      0.83      0.83       330
```

In [ ]: