Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from scipy.stats import mode

from imblearn.under_sampling import RandomUnderSampler
from sklearn.neighbors import KNeighborsClassifier
```

Read Data From The File

In [2]:
```
data=pd.read_csv("D:\\College\\College\\TERM 7\\Machine Learning\\Assignments_LABS\\Files\\Assignment 1\\magic04.data")
```

Declare feature vector and target variable

In [3]:
```
X = data.drop(['c'], axis=1)
y = data['c']
```

Balance the Datasets: (Undersampling)

In [4]:
```
# creating dataframe from X
data = pd.DataFrame(X)
# converting NDArray to series
target = pd.Series(y)

# sampling_strategy for majority class
# majority -> resamples only the majority class
undersampler = RandomUnderSampler(sampling_strategy="majority")
# resampled data and target
undersampled_data, undersampled_target = undersampler.fit_resample(data, target)
# class distribution
undersampled_target.value_counts()
```

Out[4]:
```
c
g    6688
h    6688
Name: count, dtype: int64
```

Split Dataset into Training, Validation and Test Sets

In [5]:
```
train_ratio = 0.7
validation_ratio = 0.15
test_ratio = 0.15

# train is now 70% of the entire data set
# test_size=1 - train_ratio
x_train, x_rest, y_train, y_rest = train_test_split(undersampled_data, undersampled_target, test_size=0.3)

# test is now 15% of the initial data set
# validation is now 15% of the initial data set
#test_size=test_ratio/(test_ratio + validation_ratio)
x_val, x_test, y_val, y_test = train_test_split(x_rest, y_rest, test_size=0.5)
```

Create the Model and Train it
Use Validation Set to Predict the k Value for Best Results

Steps:

1- Create an array to store our accuracy values for every k tried

2- KNeighborsClassifer function finds the k nearest neighbouring points

3- The .fit() function stores the dataset in an efficient Data Structure for searching.

4- The .predict() function classifies a data point from the validation test by finding the 'k' nearest neighbours from the Train Data and classify the query point based on the majority vote.

5- The accuracy_score() function compares the predicted value and the actual value of the new data point

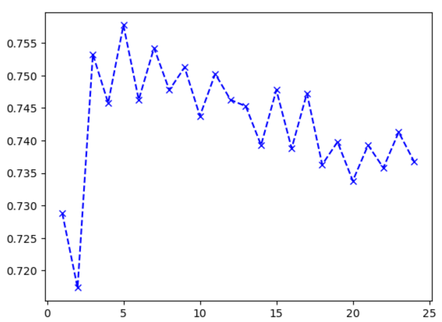
6- The accuracy is added to the accuracy list and the steps are repeated for other values of k

In [6]:
```
accuracy=[]
for i in range(1,25):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(x_train, y_train)
    y_pred = model.predict(x_val)
    acc = accuracy_score(y_val,y_pred)
    print("Accuracy: ", acc)
    accuracy.append(acc)

plt.plot(range(1,25), accuracy, color='blue', marker='x', linestyle='dashed')
```

```
Accuracy:  0.7288135593220338
Accuracy:  0.7173479561316052
Accuracy:  0.7532402791625125
Accuracy:  0.7457627118644068
Accuracy:  0.7577268195413759
Accuracy:  0.7462612163509471
Accuracy:  0.7542372881355932
Accuracy:  0.7477567298105683
Accuracy:  0.751246261216351
Accuracy:  0.7437686939182453
Accuracy:  0.7502492522432702
Accuracy:  0.7462612163509471
Accuracy:  0.7452642073778664
Accuracy:  0.7392821535393819
Accuracy:  0.7477567298105683
Accuracy:  0.7387836490528414
Accuracy:  0.747258225324028
Accuracy:  0.7362911266201396
Accuracy:  0.7397806580259222
Accuracy:  0.7337986041874377
Accuracy:  0.7392821535393819
Accuracy:  0.7357926221335992
Accuracy:  0.7412761714855434
Accuracy:  0.7367896311066799
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x2254b371b50>]
```



As the above graph shows, the best results occur at k=5 with accuracy of approximately 76%

Test the test set using the above model with k=5

In [7]:
```
model = KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
yPred=model.predict(x_test)
testDataAccuracy = accuracy_score(y_test,yPred)
print("Accuracy: ",testDataAccuracy )
```

```
Accuracy:  0.7797708021923269
```

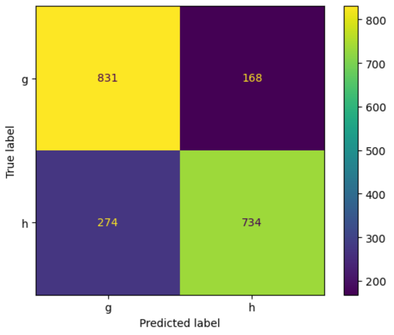
In [8]:
```
print(classification_report(y_test,yPred))
```

```
              precision    recall  f1-score   support

           g       0.75      0.83      0.79       999
           h       0.81      0.73      0.77      1008

    accuracy                           0.78      2007
   macro avg       0.78      0.78      0.78      2007
weighted avg       0.78      0.78      0.78      2007
```

In [9]:
```
ConfusionMatrixDisplay(confusion_matrix(y_test, yPred), display_labels=model.classes_).plot()
plt.show()
```



In [ ]:
```
```

In [ ]:
```
```