In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
```

In [2]:
```
data = pd.read_csv("emails.csv")
data.head()
```

Out[2]:

| | Email No. | the | to | ect | and | for | of | a | you | hou | ... | connevey | jay | valued | lay | infrastructure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Email 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| **1** | Email 2 | 8 | 13 | 24 | 6 | 6 | 2 | 102 | 1 | 27 | ... | 0 | 0 | 0 | 0 | 0 |
| **2** | Email 3 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| **3** | Email 4 | 0 | 5 | 22 | 0 | 5 | 1 | 51 | 2 | 10 | ... | 0 | 0 | 0 | 0 | 0 |
| **4** | Email 5 | 7 | 6 | 17 | 1 | 5 | 2 | 57 | 0 | 9 | ... | 0 | 0 | 0 | 0 | 0 |

5 rows × 3002 columns

In [3]:
```
df = data.copy()
df.head()
```

Out[3]:

| | Email No. | the | to | ect | and | for | of | a | you | hou | ... | connevey | jay | valued | lay | infrastructure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Email 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| **1** | Email 2 | 8 | 13 | 24 | 6 | 6 | 2 | 102 | 1 | 27 | ... | 0 | 0 | 0 | 0 | 0 |
| **2** | Email 3 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| **3** | Email 4 | 0 | 5 | 22 | 0 | 5 | 1 | 51 | 2 | 10 | ... | 0 | 0 | 0 | 0 | 0 |
| **4** | Email 5 | 7 | 6 | 17 | 1 | 5 | 2 | 57 | 0 | 9 | ... | 0 | 0 | 0 | 0 | 0 |

5 rows × 3002 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

| | the | to | ect | and | for | of | a |
|---|---|---|---|---|---|---|---|
| **count** | 5172.000000 | 5172.000000 | 5172.000000 | 5172.000000 | 5172.000000 | 5172.000000 | 5172.000000 |
| **mean** | 6.640565 | 6.188128 | 5.143852 | 3.075599 | 3.124710 | 2.627030 | 55.517401 |
| **std** | 11.745009 | 9.534576 | 14.101142 | 6.045970 | 4.680522 | 6.229845 | 87.574172 |
| **min** | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 12.000000 |
| **50%** | 3.000000 | 3.000000 | 1.000000 | 1.000000 | 2.000000 | 1.000000 | 28.000000 |
| **75%** | 8.000000 | 7.000000 | 4.000000 | 3.000000 | 4.000000 | 2.000000 | 62.250000 |
| **max** | 210.000000 | 132.000000 | 344.000000 | 89.000000 | 47.000000 | 77.000000 | 1898.000000 |

8 rows × 3001 columns

In [7]:
```
X = df.iloc[:,1:3001]
X
```

Out[7]:

| | the | to | ect | and | for | of | a | you | hou | in | ... | enhancements | connevey | jay | valued | la |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **1** | 8 | 13 | 24 | 6 | 6 | 2 | 102 | 1 | 27 | 18 | ... | 0 | 0 | 0 | 0 | |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | 4 | ... | 0 | 0 | 0 | 0 | |
| **3** | 0 | 5 | 22 | 0 | 5 | 1 | 51 | 2 | 10 | 1 | ... | 0 | 0 | 0 | 0 | |
| **4** | 7 | 6 | 17 | 1 | 5 | 2 | 57 | 0 | 9 | 3 | ... | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **5167** | 2 | 2 | 2 | 3 | 0 | 0 | 32 | 0 | 0 | 5 | ... | 0 | 0 | 0 | 0 | |
| **5168** | 35 | 27 | 11 | 2 | 6 | 5 | 151 | 4 | 3 | 23 | ... | 0 | 0 | 0 | 0 | |
| **5169** | 0 | 0 | 1 | 1 | 0 | 0 | 11 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | |
| **5170** | 2 | 7 | 1 | 0 | 2 | 1 | 28 | 2 | 0 | 8 | ... | 0 | 0 | 0 | 0 | |
| **5171** | 22 | 24 | 5 | 1 | 6 | 5 | 148 | 8 | 2 | 23 | ... | 0 | 0 | 0 | 0 | |

5172 rows × 3000 columns

In [8]:
```
Y = df.iloc[:,-1].values
Y
```

Out[8]:
```
array([0, 0, 0, ..., 1, 1, 0], dtype=int64)
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3)
```

# SVC

In [10]:
```
svc = SVC(C=1.0,kernel='rbf', gamma='auto')
svc.fit(X_train, y_train)
y_pred2 = svc.predict(X_test)
print("Accuracy Score for SVC : ", accuracy_score(y_pred2, y_test))
```

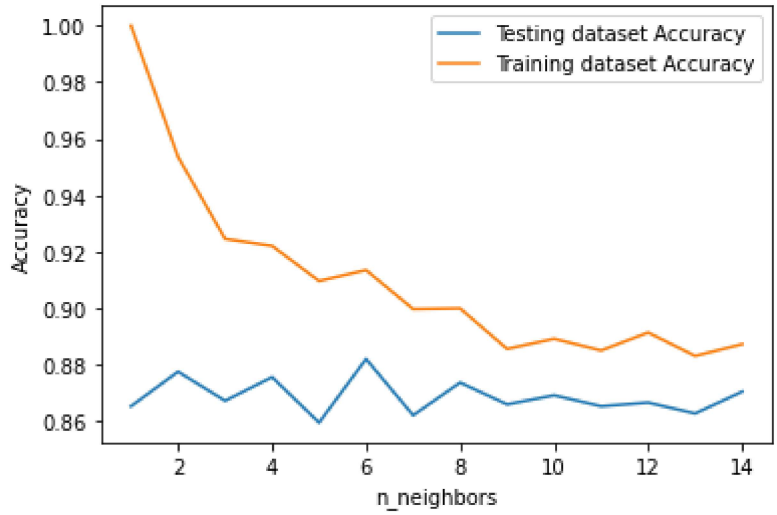
```
Accuracy Score for SVC :  0.9207474226804123
```

# KNN

In [11]:
```
knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)

knn.score(X_test, y_test)
```

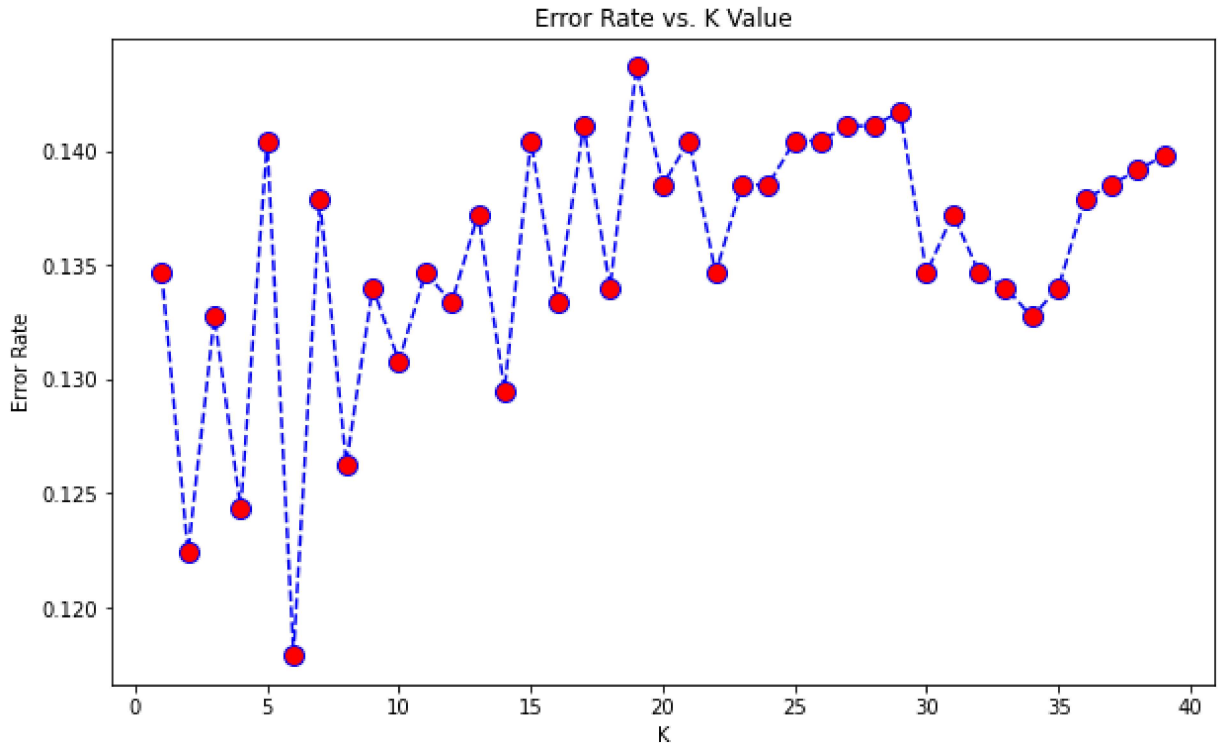
Out[11]:
```
0.8621134020618557
```

In [12]:
```
neighbors = np.arange(1, 15)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    train_accuracy[i] = knn.score(X_train, y_train)
    test_accuracy[i] = knn.score(X_test, y_test)


plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```



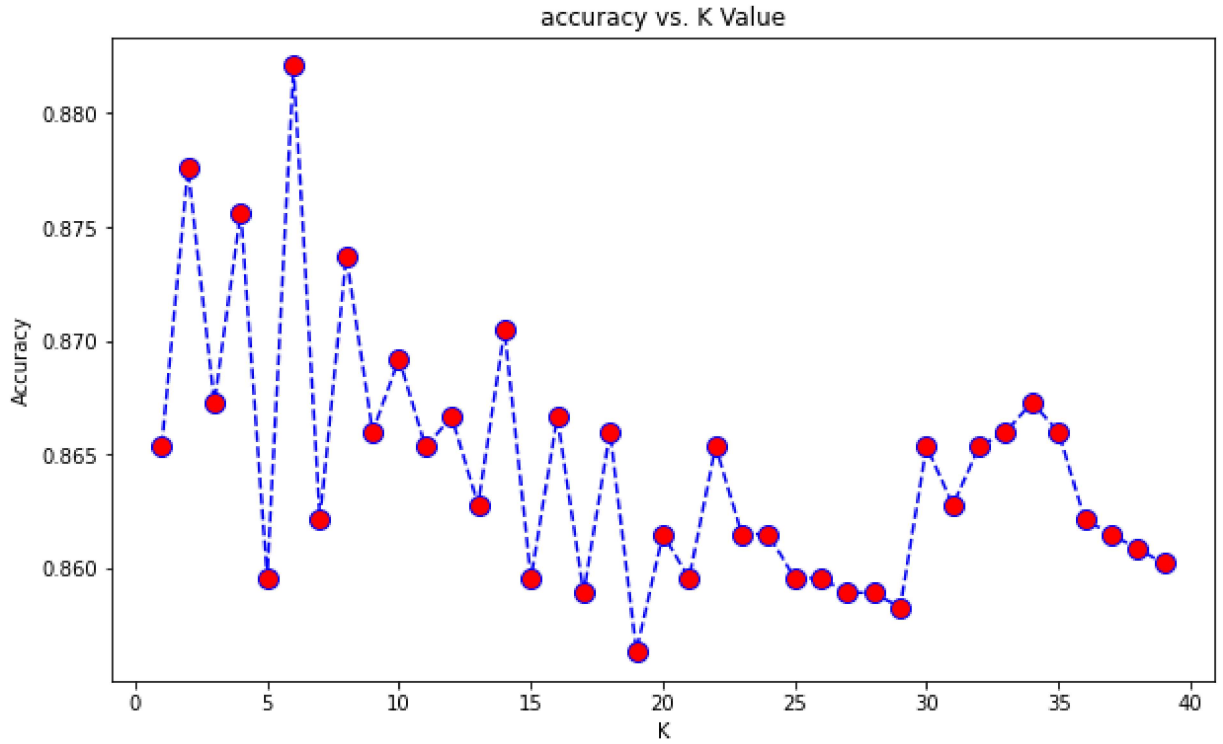
In [13]:
```
error_rate = []
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',mark
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(error_rate)
```

```
Minimum error:- 0.11791237113402062 at K = 5
```



In [14]:
```
acc = []
from sklearn import metrics
for i in range(1,40):
    neigh = KNeighborsClassifier(n_neighbors = i).fit(X_train,y_train)
    yhat = neigh.predict(X_test)
    acc.append(metrics.accuracy_score(y_test, yhat))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),acc,color = 'blue',linestyle='dashed', marker='o',markerface
plt.title('accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
print("Maximum accuracy:-",max(acc),"at K =",acc.index(max(acc)))
```

```
Maximum accuracy:- 0.8820876288659794 at K = 5
```




In [18]:
```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

knn.score(X_test, y_test)
```

Out[18]:
```
0.8595360824742269
```