In [1]:

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

```
Bad key "text.kerning_factor" on line 4 in
C:\Users\91920\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test_patch.mplstyle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.template
 (https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.templat
e)
or from the matplotlib source distribution
```

In [2]:

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

In [3]:

```
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

In [4]:

```
dataset = pd.read_csv(path, names = headernames)
print(type(dataset))
dataset.head()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[4]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
```

In [6]:

```
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,random_state=42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

In [7]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
```

Out[7]:

```
KNeighborsClassifier()
```

In [8]:

```
y_pred = classifier.predict(X_test)
```

In [9]:

```
y_pred
```

Out[9]:

```
array(['Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa'], dtype=object)
```

In [10]:

```
y_test
```

Out[10]:

```
array(['Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa'], dtype=object)
```

In [11]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,classif
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = accuracy_score(y_test,y_pred)
print("Accuracy:",result1)
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  6  1]
 [ 0  0 13]]
Accuracy: 0.9666666666666667
```

In [15]:

```
### printing the prcision,recall and other matrix
result2 = classification_report(y_test,y_pred,digits=4)
print("Classification Report:",)
print (result2)
```

```
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa     1.0000    1.0000    1.0000        10
Iris-versicolor     1.0000    0.8571    0.9231         7
 Iris-virginica     0.9286    1.0000    0.9630        13

       accuracy                         0.9667        30
      macro avg     0.9762    0.9524    0.9620        30
   weighted avg     0.9690    0.9667    0.9660        30
```

choosing different value of K

In [14]:

```
cnt =0
count=[]
train_score =[]
test_score = []
# Will take some time
for i in range(1,15):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    train_score_ = knn.score(X_train,y_train)
    test_score_ =  knn.score(X_test,y_test)
    cnt+=1
    count.append(cnt)
    train_score.append(train_score_)
    test_score.append(test_score_)

    print("for k = ", cnt)
    print("train_score is :  ", train_score_, "and test score is :  ", test_score_)
print("***********************************************")
print("***********************************************")
print("Average train score is :  ",np.mean(train_score))
print("Average test score is :  ", np.mean(test_score))
```

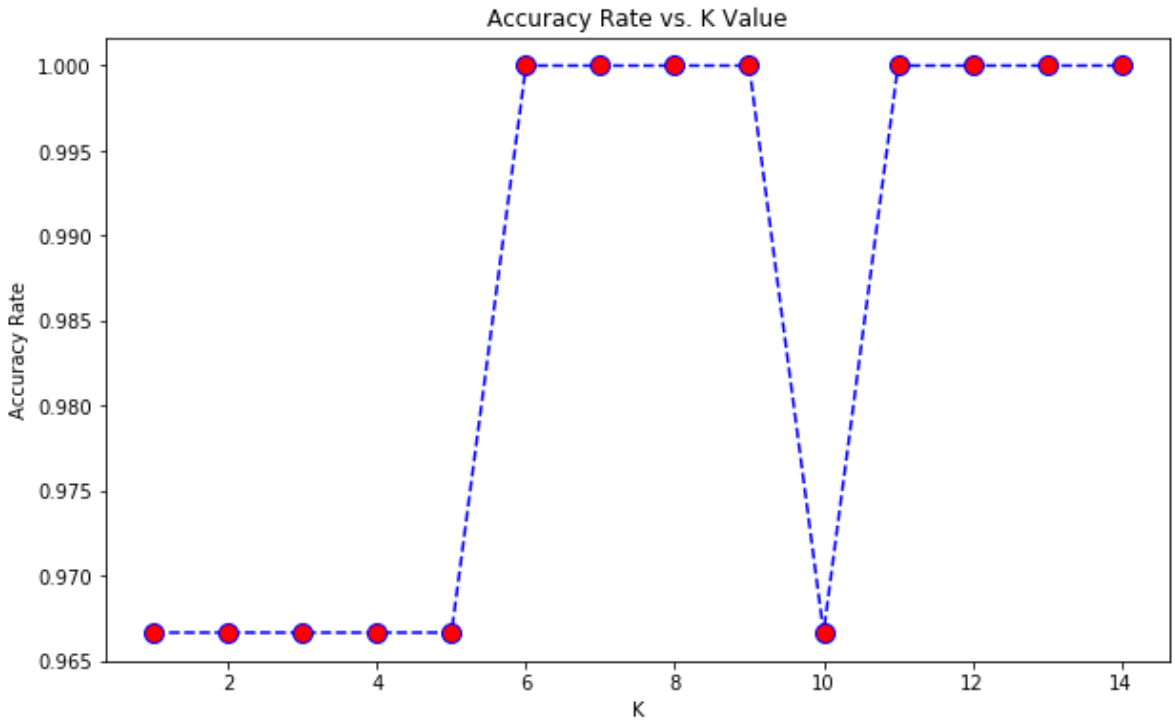
```
for k =  1
train_score is :   1.0 and test score is :   0.9666666666666667
for k =  2
train_score is :   0.975 and test score is :   0.9666666666666667
for k =  3
train_score is :   0.9666666666666667 and test score is :   0.96666666666666
67
for k =  4
train_score is :   0.9666666666666667 and test score is :   0.96666666666666
67
for k =  5
train_score is :   0.975 and test score is :   0.9666666666666667
for k =  6
train_score is :   0.9833333333333333 and test score is :   1.0
for k =  7
train_score is :   0.9833333333333333 and test score is :   1.0
for k =  8
train_score is :   0.975 and test score is :   1.0
for k =  9
train_score is :   0.975 and test score is :   1.0
for k =  10
train_score is :   0.975 and test score is :   0.9666666666666667
for k =  11
train_score is :   0.975 and test score is :   1.0
for k =  12
train_score is :   0.9833333333333333 and test score is :   1.0
for k =  13
train_score is :   0.9833333333333333 and test score is :   1.0
for k =  14
train_score is :   0.9833333333333333 and test score is :   1.0
***********************************************
***********************************************
Average train score is :   0.9785714285714283
Average test score is :   0.9857142857142858
```

In [16]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,15),test_score,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

Out[16]:

```
Text(0, 0.5, 'Accuracy Rate')
```



Hyperparameter Tuning

In [17]:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'n_neighbors' : [3,5,7,9,10,11,12,13,15,17]}
```

In [18]:

```
gridsearch = GridSearchCV(knn, param_grid,cv=10)
```

In [19]:

```
gridsearch.fit(X_train,y_train)
```

Out[19]:

```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(n_neighbors=14),
             param_grid={'n_neighbors': [3, 5, 7, 9, 10, 11, 12, 13, 15, 1
7]})
```

In [20]:

```
#df = pd.DataFrame(gridsearch.cv_results_)
```

In [21]:

```
# let's see the  best parameters according to gridsearch
gridsearch.best_params_
```

Out[21]:

```
{'n_neighbors': 5}
```

In [22]:

```
gridsearch.best_score_
```

Out[22]:

```
0.9749999999999999
```

In [ ]:

```

```

In [ ]:

```

```