# sk_kNearestNeighbor

April 15, 2019

## 1 k-Nearest Neighbor

The k-Nearest Neighbor (kNN) method makes predictions by locating similar cases to a given data instance (using a similarity function) and returning the average or majority of the most similar data instances. The kNN algorithm can be used for classification or regression.

KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consiting of training observations (x,y) and would like to capture the relationship between x and y. More formally, our goal is to learn a function h:XY so that given an unseen observation x, h(x) can confidently predict the corresponding output y

```
In [1]: from sklearn import datasets
        from sklearn import metrics
        from sklearn.neighbors import KNeighborsClassifier
        import matplotlib.pyplot as plt
        import numpy as np
```

## 2 Iris flowers Dataset

```
In [2]: dataset = datasets.load_iris()
        dataset.feature_names
```

```
Out[2]: ['sepal length (cm)',
         'sepal width (cm)',
         'petal length (cm)',
         'petal width (cm)']
```

## 3 Model

```
In [3]: model = KNeighborsClassifier()
        model.fit(dataset.data, dataset.target)
```

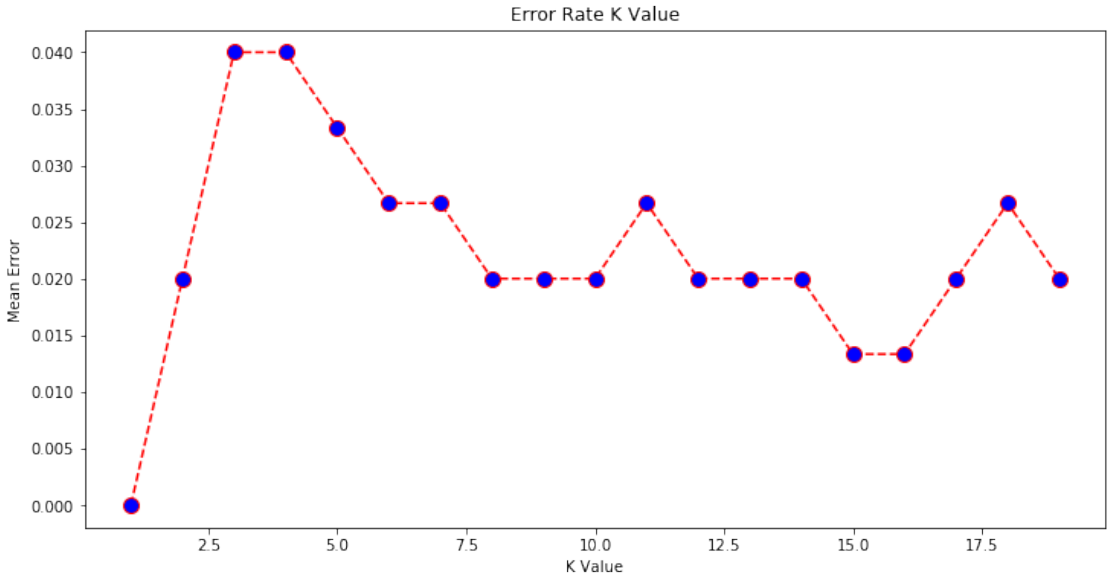
```
Out[3]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                             weights='uniform')
```

## 4 Prediction/Classification

In [4]:
```
expected = dataset.target
predicted = model.predict(dataset.data)
print("\nClassification Report : \n\n", metrics.classification_report(expected, predict
print("\nConfusion Matrix: \n\n", metrics.confusion_matrix(expected, predicted))
```

```
Classification Report :

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.96      0.94      0.95        50
           2       0.94      0.96      0.95        50

   micro avg       0.97      0.97      0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150


Confusion Matrix:

 [[50  0  0]
 [ 0 47  3]
 [ 0  2 48]]
```

In [5]:
```
# Calculating error for K values between 1 and n
n = 20
error = []
for i in range(1, n):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(dataset.data, dataset.target)
    pred_i = knn.predict(dataset.data)
    error.append(np.mean(pred_i != dataset.target))

plt.figure(figsize=(12, 6))
plt.plot(range(1, n), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
plt.show()
```




## 4.1 References

1. https://machinelearningmastery.com/get-your-hands-dirty-with-scikit-learn-now/
2. https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/
3. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/