breast_cancer_classifier

September 14, 2021

1 Breast Cancer Classifier Accuracy

In this project, we will be using breast cancer data to train a classifier of k-nearest neighbors. After, we will score our machine, and determine what the optimal k is for this project!

1.1 Imports

Our imports utilize sklearn for our dataset/analysis and matplotlib for graphics visualization.

[2]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
```

First lets import our data and take a quick look at the set.

[3]:
```
breast_cancer_data = load_breast_cancer()
print('Overall Data:')
print(breast_cancer_data.data[0])
print(f'Feature names:\n{breast_cancer_data.feature_names}')
print(f'Data target:\n{breast_cancer_data.target}')
print(f'Target names:\n{breast_cancer_data.target_names}')
```

```
Overall Data:
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
Feature names:
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
Data target:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
Target names:
['malignant' 'benign']
```

1.2 Building The Model

Now we can define our training and testing set. Let's change our test size to 20% from the default 25%.

[4]:
```
training_data, validation_data, training_labels, validation_labels =␣
↪train_test_split(breast_cancer_data.data, breast_cancer_data.target,␣
↪test_size=0.2, random_state=100)
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(training_data, training_labels)
```

[4]:
```
KNeighborsClassifier(n_neighbors=3)
```

Now we can look at hour our classifier scored with 3 neighbors.

[5]:
```
print(classifier.score(validation_data, validation_labels))
```

```
0.9473684210526315
```

1.3 Improving The Model

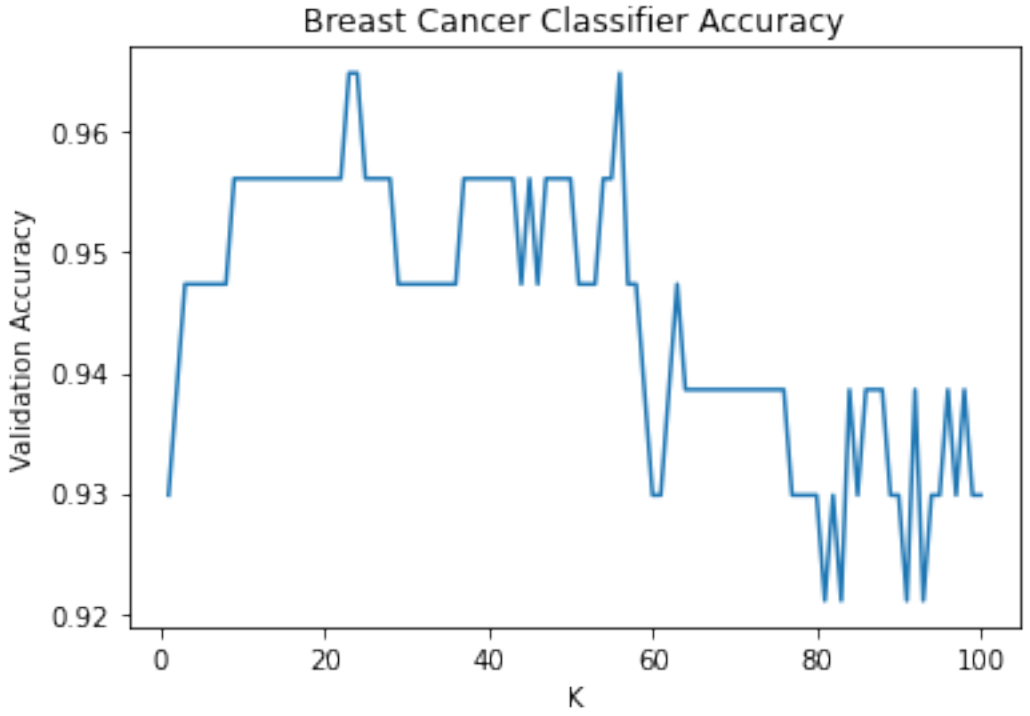
94% is excellent, but let's see if we can't improve it further by testing different neighbor (k) values.

[6]:
```
accuracies = []
for k in range(1, 101):
  classifier = KNeighborsClassifier(n_neighbors=k)
  classifier.fit(training_data, training_labels)
  accuracies.append(classifier.score(validation_data, validation_labels))
```

```
k_list = range(1, 101)
```

Let's plot our k's and their accuracies.

[7]:
```
plt.plot(k_list, accuracies)
plt.xlabel('K')
plt.ylabel('Validation Accuracy')
plt.title('Breast Cancer Classifier Accuracy')
plt.show()
```



And finally, let's grab the best value from our analysis.

[8]:
```
# creating a dictionary of our k's and accuracies
k_dict = dict(zip(k_list, accuracies))
# printing out the max accuracy and it's k
max_score = max(k_dict.values())
print(f'The best score ({max_score}) comes from a classifier with: {max(k_dict,␣
↪key=k_dict.get)} neighbors!')
```

```
The best score (0.9649122807017544) comes from a classifier with: 23 neighbors!
```

From 94% to 96%! That's a pretty good initial improvement!

Data Sources Data was provided by the sklearn package.