knn-class

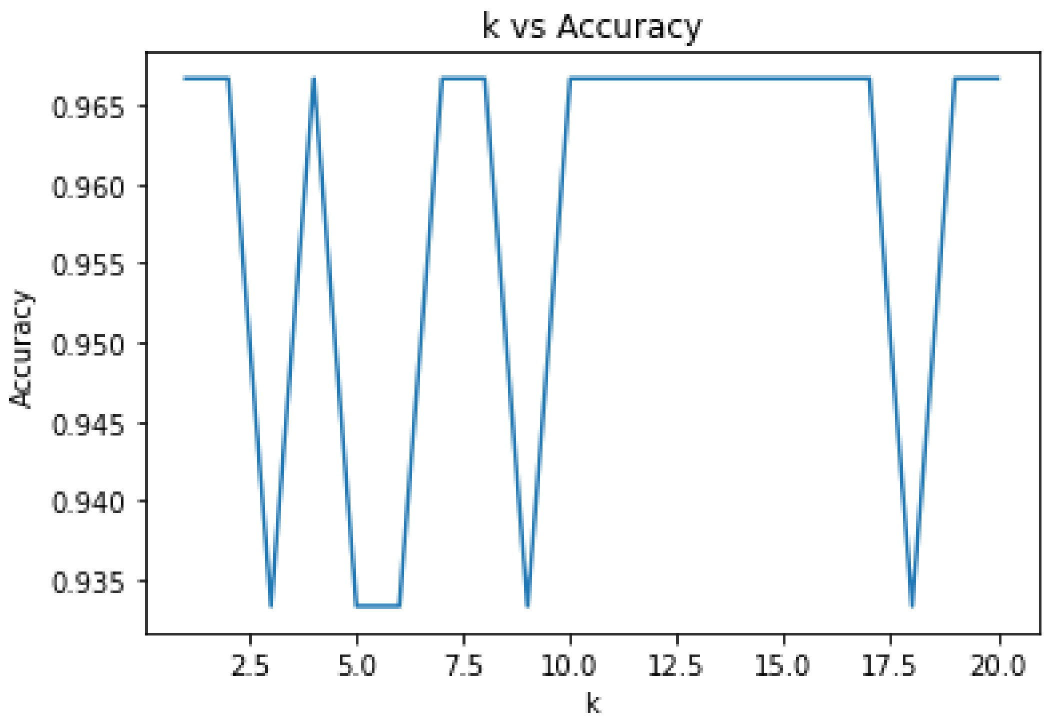
March 20, 2024

[8]:

```python
import numpy as np

# Define Euclidean distance function
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

# Define K-NN classifier function
def knn(train_data, train_labels, test_instance, k):
    distances = []
    for i in range(len(train_data)):
        dist = euclidean_distance(test_instance, train_data[i])
        distances.append((train_labels[i], dist))
    distances.sort(key=lambda x: x[1])
    neighbors = distances[:k]
    class_votes = {}
    for neighbor in neighbors:
        label = neighbor[0]
        if label in class_votes:
            class_votes[label] += 1
        else:
            class_votes[label] = 1
    sorted_votes = sorted(class_votes.items(), key=lambda x: x[1], reverse=True)
    return sorted_votes[0][0]

# Define function to calculate accuracy
def accuracy(y_true, y_pred):
    correct = 0
    for i in range(len(y_true)):
        if y_true[i] == y_pred[i]:
            correct += 1
    return correct / len(y_true)

# Define function to split dataset into train and test sets
def train_test_split(data, labels, test_size=0.2):
    data_size = len(data)
    test_data_size = int(data_size * test_size)
    indices = np.random.permutation(data_size)
```

```
    test_indices = indices[:test_data_size]
    train_indices = indices[test_data_size:]
    train_data = data[train_indices]
    train_labels = labels[train_indices]
    test_data = data[test_indices]
    test_labels = labels[test_indices]
    return train_data, test_data, train_labels, test_labels

# Define function to plot k vs accuracy curve
def plot_k_vs_accuracy(train_data, train_labels, test_data, test_labels):
    k_values = range(1, 21)
    accuracies = []
    for k in k_values:
        y_pred = []
        for instance in test_data:
            y_pred.append(knn(train_data, train_labels, instance, k))
        accuracies.append(accuracy(test_labels, y_pred))
    plt.plot(k_values, accuracies)
    plt.xlabel('k')
    plt.ylabel('Accuracy')
    plt.title('k vs Accuracy')
    plt.show()
    return k_values, accuracies

# Define main function
def main():
    # Load dataset (example: Iris dataset)
    from sklearn.datasets import load_iris
    iris = load_iris()
    data = iris.data
    labels = iris.target

    # Split dataset into train and test sets
    train_data, test_data, train_labels, test_labels = train_test_split(data,␣
↪labels)

    # Plot k vs accuracy curve
    k_values, accuracies = plot_k_vs_accuracy(train_data, train_labels,␣
↪test_data, test_labels)

    # Find k for maximum accuracy
    max_accuracy = max(accuracies)
    max_accuracy_k = k_values[accuracies.index(max_accuracy)]
    print(f'Maximum accuracy: {max_accuracy:.2f} for k = {max_accuracy_k}')

    # Test model using test set and find accuracy and confusion matrix for best␣
↪k
```

```python
    y_pred = []
    for instance in test_data:
        y_pred.append(knn(train_data, train_labels, instance, max_accuracy_k))
    print(f'Confusion matrix:\n{confusion_matrix(test_labels, y_pred)}')
    print(f'Accuracy: {accuracy(test_labels, y_pred):.2f}')

if __name__ == '__main__':
    main()
```




```
Maximum accuracy: 0.97 for k = 1
Confusion matrix:
[[10  0  0]
 [ 0 11  1]
 [ 0  0  8]]
Accuracy: 0.97
```