Normal KNN

In [4]:

```
from collections import Counter
import numpy as np
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        y_pred = [self._predict(x) for x in X]
        return np.array(y_pred)

    def _predict(self, x):
        # Compute distances between x and all examples in the training set
        distances = [euclidean_distance(x, x_train) for x_train in self.X_train]
        # Sort by distance and return indices of the first k neighbors
        k_idx = np.argsort(distances)[: self.k]
        # Extract the labels of the k nearest neighbor training samples
        k_neighbor_labels = [self.y_train[i] for i in k_idx]
        # return the most common class label
        most_common = Counter(k_neighbor_labels).most_common(1)
        return most_common[0][0]


if __name__ == "__main__":
    # Imports
    from matplotlib.colors import ListedColormap
    from sklearn import datasets
    from sklearn.model_selection import train_test_split

    cmap = ListedColormap(["#FF0000", "#00FF00", "#0000FF"])

    def accuracy(y_true, y_pred):
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy
```

```
iris = datasets.load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)

k = 3
clf = KNN(k=k)
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
print("KNN classification accuracy", accuracy(y_test, predictions))
```

KNN classification accuracy 1.0

Parallel KNN

In [5]:

```
from collections import Counter
import numpy as np
from joblib import Parallel, delayed
from sklearn.metrics import pairwise_distances

class KNN:
    def __init__(self, k=3, n_jobs=1):
        self.k = k
        self.n_jobs = n_jobs

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        dists = pairwise_distances(X, self.X_train, n_jobs=self.n_jobs)
        indices = np.argsort(dists, axis=1)[:, :self.k]
        y_pred = Parallel(n_jobs=self.n_jobs)(delayed(self._predict)(indices[i]) for i in
range(X.shape[0]))
        return np.array(y_pred)

    def _predict(self, indices):
        k_neighbor_labels = self.y_train[indices]
        most_common = Counter(k_neighbor_labels).most_common(1)
        return most_common[0][0]
```

```
if __name__ == "__main__":
    # Imports
    from matplotlib.colors import ListedColormap
    from sklearn import datasets
    from sklearn.model_selection import train_test_split

    cmap = ListedColormap(["#FF0000", "#00FF00", "#0000FF"])

    def accuracy(y_true, y_pred):
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy

    iris = datasets.load_iris()
    X, y = iris.data, iris.target

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)

    k = 3
    n_jobs = 4  # Number of parallel jobs

    clf = KNN(k=k, n_jobs=n_jobs)
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    print("KNN classification accuracy:", accuracy(y_test, predictions))
```

```
KNN classification accuracy: 1.0
```

In [ ]: