Rapport KNN

J'ai remarqué que le comportement de l'algorithme peut être impacter par k(le nombre de voisin), la méthode choisit pour le vote (k commun ou k plus grand, etc.) et le ratio entre les données d'apprentissage et les données de test.

```
def distance(point_1, point_2):
    """
    Distance entre deux points
    point_1 = [a, b, c, d, e]
    point_2 = [a', b', c', d', e']
    distance = sqrt((a - a')**2 + ...)
    """
    return np.sqrt(sum((point_1[i] - point_2[i])**2 for i in range(len(point_1))))

def plus_commun(liste):
    return max(set(liste), key=liste.count)

def obtenir_k_voisins(point):
    """
        Une condition pour que l'algo s'adapte automatiquement aux données.
        Deux types de données :
            - Données de test (on a la classe)
            - Données à prédire (pas de classe)
    """
    if len(point) == 7:
        cle = lambda p: distance(p[:-1], point[:-1])
    else:
        cle = lambda p: distance(p[:-1], point)
    donnes_triees = sorted(donnees_apprentissage, key=cle)
    voisins = donnes_triees[:constante_k]
    return voisins

def predire(point):
    voisins = obtenir_k_voisins(point)
    classes = [v[-1] for v in voisins]
    return plus_commun(classes)
```

La méthode utilisée dans cet algorithme est la distance euclidienne. L'algorithme mesure les distances entre les points et nous renvoie les k plus proches voisins. Ensuite l'algo prend la décision selon la répétition d'une classe.

Exemple : Les k plus proches sont {A, B, C, C, C}. Ici, on a les 5 plus proches voisins, « C » est répété 3 fois, la prédiction est donc C.

Le choix de K après plusieurs tests et mesures est important, car si un k trop petit, l'algorithme peut se tromper. Si k est trop grand, l'algorithme peut se tromper dans la décision. Dans le graphique ci-dessous, on remarque que le nombre d'erreur augmente selon le nombre de ses voisins.

L'algorithme est optimal à k=3.

Avec un k = 3 et un ratio 90/10 entre les données d'apprentissage/test, on obtient 90% de précision.