K Nearest Neighbors

مقدمه :

فرض کنید جانوران را به دو دسته تقسیم بندی کرده ایم. دسته اول جانورانی که در آب زندگی می کنند و آن ها را به عنوان آبزی می شناسیم ، دسته دوم جانورانی که در خشکی زندگی می کنند و آن ها را به عنوان غیر آبزی می شناسیم. اگر جانور جدیدی را خارج از آب مشاهده کنیم و از ما سوال بپرسند که این جانور آبزی است یا خیر ، ما پاسخ می دهیم که این جانور غیر آبزی است.

K Nearest Neighbors چیست؟

ابتدا داده ها را به دو قسمت Train و Test تقسیم بندی می کنیم. تصور کنید می خواهیم با استفاده از الگوریتم KNN تشخیص دهیم که میوه ای با ویژگی های معلوم ، چه نوع میوه ای است. داده های Train شامل انواع میوه ها می شود که ویژگی ها و نوع آن ها مشخص است. ویژگی های میوه ها شامل وزن ، ارتفاع ، عرض و عددی مربوط به رنگ میوه می شود. بر اساس داده های آموزش ، الگوریتم KNN یادگیری خود را انجام می دهد. فرآیند یادگیری بدین شکل است که بر اساس ویژگی هایی که وجود دارد ، داده ها را به چند بخش تقسیم می کند برای مثال به چند نوع میوه (سیب ، لیمو و...). حال به عنوان ورودی داده های تست را به الگوریتم می دهیم. بر اساس ویژگی های مشخص شده الگوریتم KNN به دنبال نزدیکترین همسایه می گردد و نتیجه نزدیکترین همسایه را برای ورودی نیز قرار می دهد. برای مثال اگر ابعاد ورودی ، اعدادی نزدیک به ابعاد میوه سیب باشد ، نوع میوه را برای ورودی سیب در نظر می گیرد. در نهایت خروجی مربوط به داده های تست که به واسطه الگوریتم مشخص شده است با خروجی اصلی آن ها که در دیتاست آمده است مقایسه می شود و به عنوان دقت عملکرد یا به عنوان خطای الگوریتم در نظر گرفته می شود.

نکته ! K به عنوان هایپر پارامتر برای این الگوریتم است. یعنی شما باید مقدار آن را مشخص کنید و منظور از آن ، تعداد همسایه های اطراف برای بررسی است. برای مثال اگر مقدار آن را ۳ در نظر بگیرید در بین دیتاهای اطراف ، ۳ داده ای که از نظر مقادیر ویژگی نزدیکترین مقادیر را دارند در نظر گرفته و Label این ۳ را بررسی می کند. هر Label که تعدادش بیشتر باشد به عنوان Label برای داده ورودی در نظر گرفته می شود. تغییر مقدار K می تواند خروجی و نتیجه را تغییر دهد.

نکته ! برای پیداکردن نزدیکترین همسایه باید فاصله بین مقادیر ویژگی ها محاسبه شود. معمولا از فاصله اقلیدسی برای محاسبه استفاده می شود.

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

در اینجا x و y به عنوان ویژگی های مختلف در نظر گرفته می شود که برای دو داده مختلف است.

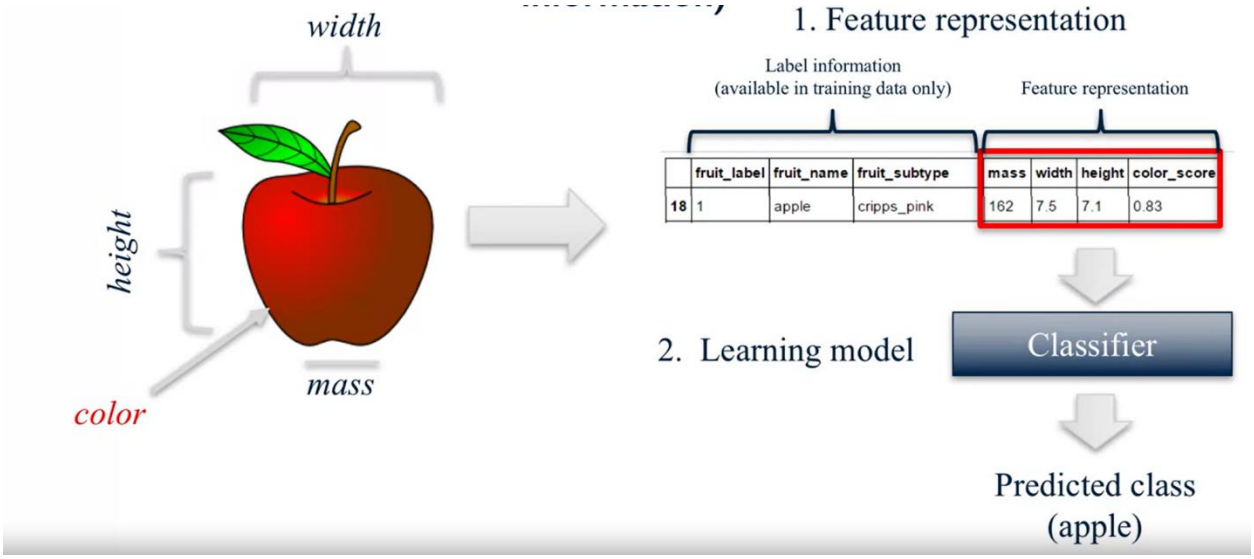
نکته ! بهتر است مقدار K عددی فرد انتخاب شود زیرا اگر حالت تساوی رخ دهد آنگاه دچار مشکل خواهیم شد. (البته الگوریتم های تساوی شکن در پایتون وجود دارد اما بهتر است فرد انتخاب کنیم)

نکته ! هر چه مقدار K بیشتر باشد احتمالا دقت بیشتر می شود.

نکته ! اگر تعداد داده ها زیاد باشد ، الگوریتم KNN هزینه محاسباتی زیادی خواهد داشت.

پروژه تشخیص میوه :

می خواهیم با استفاده از دیتاست مربوط به میوه ها و با استفاده از الگوریتم KNN برنامه ای بنویسیم که ویژگی های مربوط به میوه ای را به عنوان داده جدید دریافت کند و تشخیص دهد که میوه ، چه نوع میوه ای است.



برای حل این پروژه ابتدا کتابخانه های مورد نیاز و دیتاست را فراخوانی می کنیم.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

fruits = pd.read_csv('fruit_data_with_colours.csv')
fruits.head()
```

انواع میوه ای که در دیتاست وجود دارد به همراه ID که برای آن در دیتاست در نظر گرفته شده است را برای کسب اطلاعات بیشتر درباره دیتاست مشاهده می کنیم.

```
lookup_fruit_name = dict(zip(fruits.fruit_label.unique(), fruits.fruit_name.unique()))
print(lookup_fruit_name)
```

دیتاست را به دو دسته ی داده های آموزش و داده های تست تقسیم بندی می کنیم.

```
from sklearn.model_selection import train_test_split

X = fruits[['height', 'width', 'mass', 'color_score']]
y = fruits['fruit_label']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

نمودار مربوط به مقادیر ویژگی ها را رسم می کنیم تا دید بهتری نسبت به روند انجام آموزش پیدا کنیم.

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(X_train['width'], X_train['height'], X_train['color_score'], c = y_train, marker = 'o', s=100)
ax.set_xlabel('width')
ax.set_ylabel('height')
ax.set_zlabel('color_score')
plt.show()
```

برای پیاده سازی الگوریتم KNN نیاز به ساخت Object از این ماژول داریم.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 5)
```

حال باید الگوریتم پیاده سازی شده را آموزش دهیم.

```
knn.fit(X_train, y_train)
```

برای بدست آوردن دقت عملکرد الگوریتم داریم :

```
knn.score(X_test, y_test)
```

حال از الگوریتم آموزش داده شده ، برای پیش بینی داده جدید استفاده می کنیم.

```
fruit_prediction = knn.predict([[20, 4.3, 5.5, 0.795]])
lookup_fruit_name[fruit_prediction[0]]
```

نکته ! الگوریتم های طبقه بندی می توانند خروجی شامل دو کلاس (Binary Classification) و یا چند کلاس (Multiple Classification) داشته باشند.