مقدمه:

الگوریتم KNN (k نزدیکترین همسایه - K-nearest neighbors) نوعی از الگوریتم های یادگیری ماشین تحت نظارت است که هم در مسائل طبقه بندی و هم در مسائل رگرسیون پیشگویانه مورد استفاده قرار می گیرد. اگرچه، غالبا در مسائل طبقه بندی پیشگویانه، در صنعت از آن استفاده می شود. دو خصیصه زیر KNN را به خوبی تعریف می کند:

الگوریتم یادگیری تنبل (Lazy learning algorithm) : KNN یک الگوریتم یادگیری تنبل است زیرا یک مرحله خاص جهت یادگیری ندارد، و در هنگام طبقه بندی از تمام داده برای یادگیری استفاده می کند.

الگوریتم یادگیری بدون پارامتر (Non-parametric learning algorithm) : همچنین KNN یک الگوریتم یادگیری بدون پارامتر است زیرا درباره داده اصلی هیچ فرضی در نظر نمی گیرد.

عملکرد الگوریتم KNN : الگوریتم KNN (k نزدیکترین همسایه - K-nearest neighbors) از "تشابه ویژگی" برای پیش بینی مقادیر نقاط داده جدید استفاده می کند؛ که به این معنی است که به نقطه داده جدید بر اساس میزان مطابقت آن با نقاط مجموعه آموزشی، یک مقدار تخصیص می دهد. با کمک مراحل زیر میتوان نحوه عملکرد آن را درک کرد.

مرحله ۱- برای پیاده سازی هر الگوریتمی، به مجموعه داده نیاز داریم. بنابراین، در طول مرحله اول از KNN، باید داده آموزشی را به همراه داده تست بارگیری کنیم.

مرحله ۲- سپس، باید مقدار k را انتخاب کنیم مانند نزدیک ترین نقاط داده. K می تواند هر عدد صحیحی باشد.

مرحله ۳- برای هر نقطه داده در مجموعه داده تست، مراحل زیر را انجام دهید.

۳,۱ با کمک هر یک از متد های اقلیدسی (Euclidean) ، فاصله همینگ (Hamming distance) یا منهتن (Manhattan distance) ، فاصله بین داده تست و هر سطر از داده آموزشی را محاسبه کنید. متداول ترین روش برای محاسبه فاصله، روش اقلیدسی است.

۳,۲ حال، بر اساس مقدار فاصله، آنها را به صورت صعودی مرتب کنید.

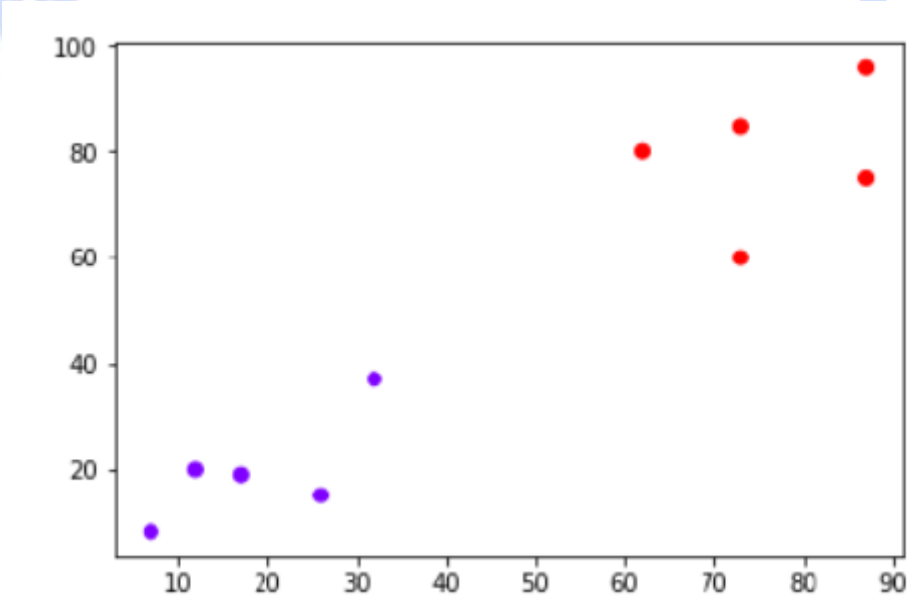
۳,۳ سپس، الگوریتم، k سطر بالاتر از آرایه مرتب شده را انتخاب میکند.

۳,۴ حال، بر اساس متداول ترین کلاس از این سطر ها، الگوریتم یک کلاس به نقطه تست تخصیص می دهد.

مرحله ۴ – پایان.

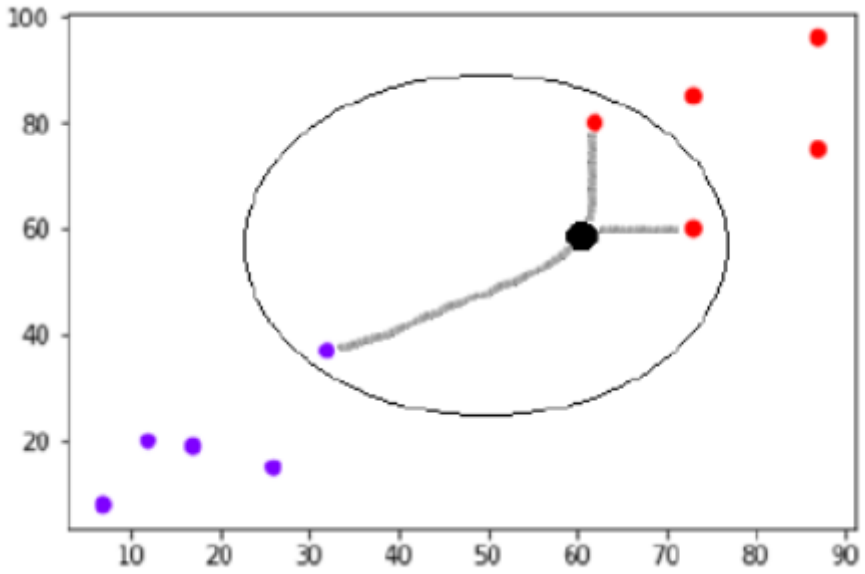
مثال: در ادامه مثالی برای درک مفهوم k و نحوه عملکرد الگوریتم KNN ارائه شده است.

فرض کنید یک مجموعه داده داریم که می توان آن را به صورت زیر رسم کرد.



حال، باید نقطه داده جدید به رنگ مشکی (در مختصات ۶۰،۶۰) را در کلاس آبی یا قرمز طبقه بندی کنیم.

فرض می کنیم k=3 است در نتیجه سه تا از نزدیک ترین نقاط داده را پیدا میکند که در شکل زیر نشان داده شده است.



در شکل فوق می توان سه تا از نزدیکترین همسایگان نقطه داده سیاه رنگ را دید. در بین آن سه تا، دو تا از آنها در کلاس قرمز هستند در نتیجه نقطه سیاه نیز به کلاس قرمز تخصیص می یابد.

پیاده سازی در پایتون: همانطور که می دانیم از الگوریتم KNN می توان هم برای طبقه بندی و هم رگرسیون استفاده کرد. در ادامه روش هایی در پایتون جهت استفاده از KNN به عنوان طبقه بندی کننده و رگرسر معرفی می شود.

KNN به عنوان طبقه بندی کننده (classifier): ابتدا، با وارد کردن بسته های ضروری شروع می کنیم.

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

سپس، با استفاده از لینک زیر مجموعه داده iris را دانلود کنید.

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

سپس، باید به صورت زیر اسامی ستون ها را به مجموعه داده تخصیص داد.

```
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

حال، باید مجموعه داده را در قالب داده pandas به روش زیر بخوانیم.

```
dataset = pd.read_csv(path, names = headernames)
dataset.head()
```

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

با کمک کد زیر، پیش پردازش داده انجام خواهد شد.

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
```

سپس، داده را به دو بخش آموزشی و تست تقسیم می کنیم. کد زیر مجموعه داده را به گونه ای تقسیم می کند که ۶۰ درصد داده آموزشی و ۴۰ درصد داده تست داشته باشیم.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40)
```

سپس، بازسازی داده (data scaling) به صورت زیر انجام می شود.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

سپس، با کمک کلاس KNeighborsClassifier از sklearn ، مدل را به صورت زیر آموزش دهید.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 8)
```

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

در انتها با کمک اسکریپت زیر باید پیش بینی کنیم.

```
y_pred = classifier.predict(X_test)
```

سپس، نتایج را به صورت زیر چاپ کنید.

```
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

خروجی:

```
Confusion Matrix:
[[21  0  0]
 [ 0 16  0]
 [ 0  7 16]]
Classification Report:
                 precision    recall  f1-score
support
    Iris-setosa       1.00      1.00      1.00
21
Iris-versicolor       0.70      1.00      0.82
16
 Iris-virginica       1.00      0.70      0.82
23
      micro avg       0.88      0.88      0.88
60
      macro avg       0.90      0.90      0.88
60
   weighted avg       0.92      0.88      0.88
60

Accuracy: 0.8833333333333333
```

KNN به عنوان رگرسر (regressor): ابتدا با وارد کردن بسته های ضروری پایتون شروع می کنیم.

```
import numpy as np
```

```
import pandas as pd
```

سپس، مجموعه داده iris را از لینک زیر دانلود کنید.

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

سپس، باید به صورت زیر اسامی ستون ها را به مجموعه داده تخصیص دهیم.

```
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

حال، باید به صورت زیر، مجموعه داده را در قالب داده pandas بخوانیم.

```
data = pd.read_csv(url, names = headernames)
array = data.values
X = array[:,:2]
Y = array[:,2]
data.shape
output:(150, 5)
```

سپس، *KNeighborsRegressor* از *sklearn* را برای متناسب کردن مدل وارد کنید.

```
from sklearn.neighbors import KNeighborsRegressor
knnr = KNeighborsRegressor(n_neighbors = 10)
knnr.fit(X, y)
```

در انتها می توان به صورت زیر MSE را پیدا کرد.

```
print ("The MSE is:",format(np.power(y-knnr.predict(X),2).mean()))
```

خروجی:

```
The MSE is: 0.12226666666666669
```

مزایا و معایب KNN :

مزایا: الگوریتمی بسیار ساده برای فهم و تفسیر است. از آنجایی که در این الگوریتم هیچ فرضی درباره داده نمی شود، برای داده های غیر خطی بسیار مناسب است. از آنجایی که می توان آن را هم برای طبقه بندی و هم برای رگرسیون استفاده کرد، یک الگوریتم همه کاره است. نسبتا دارای دقت بالایی است اما مدل های یادگیری تحت نظارت بسیار بهتری از KNN نیز وجود دارد.

معایب: از آنجایی که همه داده آموزشی را ذخیره می کند، از لحاظ محاسباتی کمی گران است. در مقایسه با سایر الگوریتم های یادگیری تحت نظارت نیازمند حافظه ذخیره سازی بالایی است. در صورت داشتن K بزرگ، پیش بینی آهسته خواهد بود. نسبت به مقیاس داده و نیز ویژگی های نا مربوط بسیار حساس است.

برنامه های کاربردی KNN : موارد زیر برخی از زمینه هایی است که KNN با موفقیت در آن ها اعمال می شود.

سیستم بانکی: در یک سیستم بانکی با استفاده از KNN می توان پیش بینی کرد که آیا یک فرد برای وام مناسب است یا خیر. آیا آن فرد دارای خصوصیات مشابه با متخلفین هست یا خیر.

محاسبه مرتبه اعتبار: می توان از الگوریتم KNN برای یافتن مرتبه اعتبار افراد با مقایسه آنها با افرادی که دارای خصوصیات مشابه هستند استفاده کرد.

سیاست: با کمک الگوریتم های KNN، می توانیم یک رای دهنده بالقوه را بین کلاس های متنوع مانند "رای می دهد" ، "رای نمی دهد" ، به حزب کنگره رای می دهد" و "به حرب BJP " رای می دهد، طبقه بندی کنیم.

سایر زمینه هایی که می توان از الگوریتم KNN در آنها استفاده کرد عبارتند از تشخیص صدا، تشخیص دست خط، تشخیص تصویر و ویدئو.