

دوره دیتا ساینس کاربردی

Machine Learning

K Nearest Neighbors

x x
x x
x x
x x

x x
x x
x x
x x

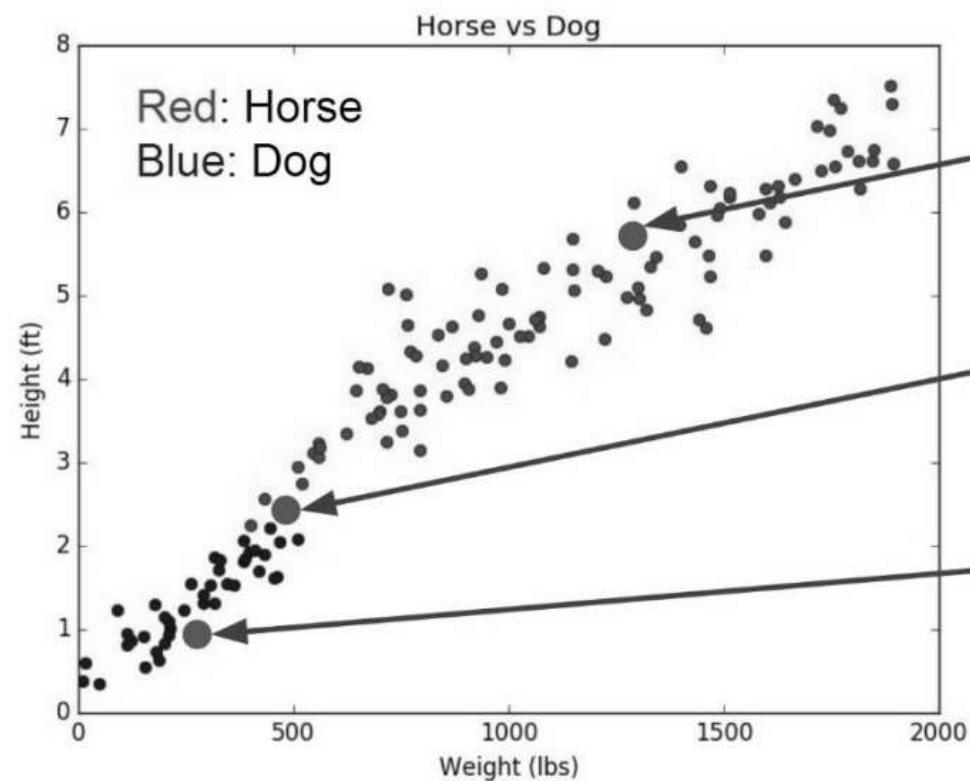
—● dataroadmap ●—

مدرس: مونا حاتمی



جلسه هفتم

K Nearest Neighbors (KNN) Supervised Learning

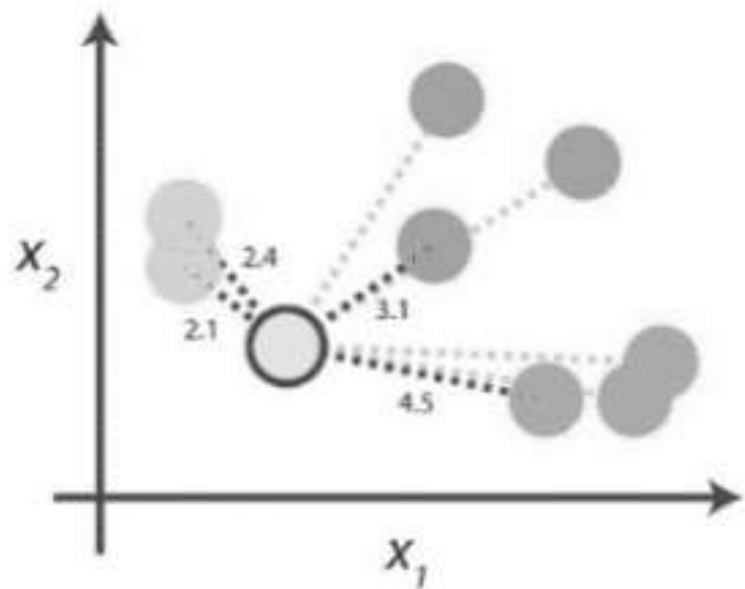










New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

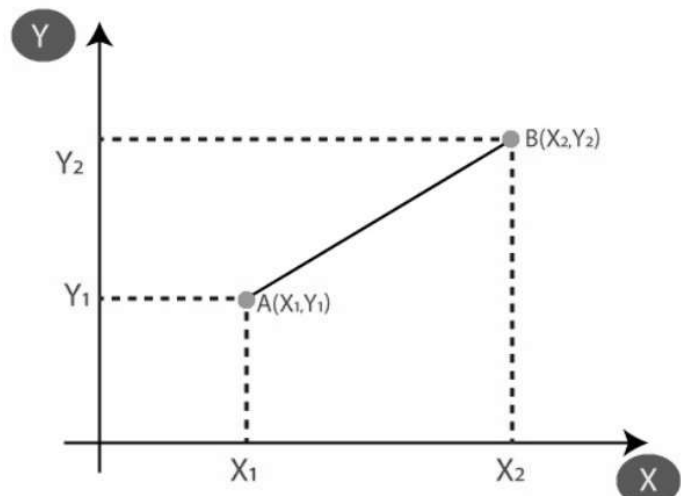
New datapoint:
Is it a horse or a dog?

K Nearest Neighbors (KNN)

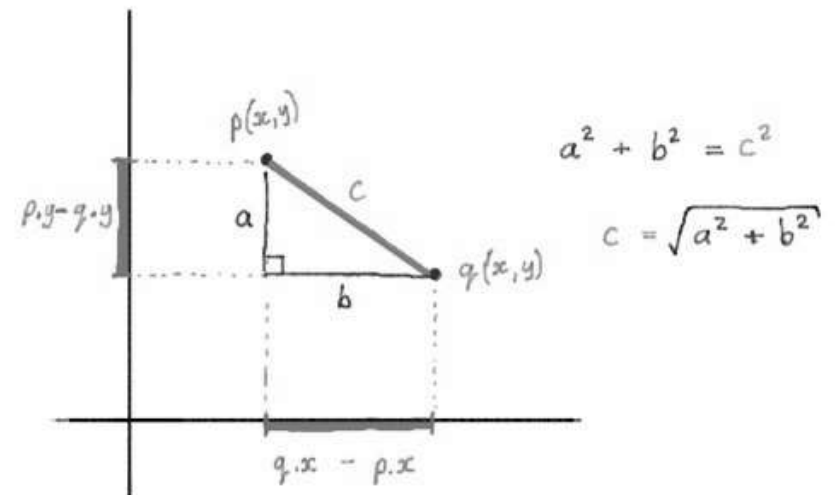


Point Distance			
	...		2.1 → 1st NN
	...		2.4 → 2nd NN
	...		3.1 → 3rd NN
	...		4.5 → 4th NN

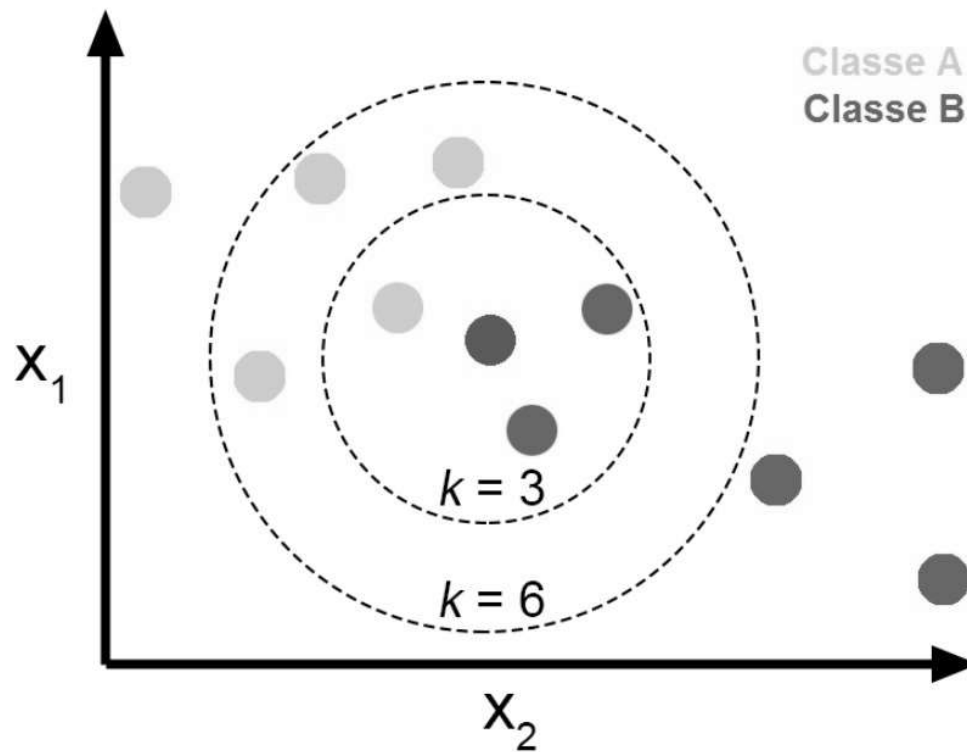
Euclidean Distance



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$



KNN, K Number



KNN, Pros & Cons

مزایای این روش:

ساده است.

چندین دسته را دربر میگیرد.

معایب این روش:

برای دیتاست های بزرگ خیلی هزینه بر است.

روش نامناسب برای دیتاست هایی با پارامتر زیاد است.

Read the Data

```
import pandas as pd
```

The Data ¶

```
df = pd.read_csv('preprocessed_dataset.csv')
```

```
df.head()
```

[3]:

	Unnamed: 0	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCo
0	0	6104.959412	1	0	0	0	1.0	
1	1	525.000000	1	0	0	0	1.0	
2	2	677.000000	1	0	0	0	1.0	

Non- Null

```
|: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 90 entries, 0 to 89
```

```
Data columns (total 89 columns):
```

#	Column	Non-Null Count	Dtype
0	Unnamed: 0	90 non-null	int64
1	PayloadMass	90 non-null	float64
2	Flights	90 non-null	int64
3	GridFins	90 non-null	int64
4	Reused	90 non-null	int64
5	Legs	90 non-null	int64
6	Block	90 non-null	float64
7	ReusedCount	90 non-null	int64
8	Class	90 non-null	int64
9	Orbit_ES-L1	90 non-null	int64
10	Orbit_GEO	90 non-null	int64
11	Orbit_GTO	90 non-null	int64
12	Orbit_HEO	90 non-null	int64
13	Orbit_ISS	90 non-null	int64
14	Orbit_LEO	90 non-null	int64

Define X , y

```
X=df.drop('Class',axis=1)  
y=df['Class']
```

Train set- Test set in sklearn library

```
! pip install sklearn
```

```
from sklearn.model_selection import train_test_split
```

[illegible]

Train the model

```
In [8]: ➤ # K Nearest Neighbors classification algorithm  
from sklearn.neighbors import KNeighborsClassifier
```

```
In [9]: ➤ knn = KNeighborsClassifier(n_neighbors=10)
```

```
In [10]: ➤ knn.fit(X_train,y_train)
```

```
Out[10]: KNeighborsClassifier(n_neighbors=10)
```

Prediction

VS

y_test

```
In [11]: ► predictions = knn.predict(X_test)
```

```
In [12]: ► predictions
```

```
Out[12]: array([0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

```
In [13]: ► y_test
```

```
Out[13]: 50    0
          6    1
          51   0
          54   1
          53   1
          69   1
          32   1
          31   1
          21   1
          88   1
          43   1
          47   0
           3   0
           1   0
          74   0
          16   1
          45   0
          25   1
          Name: Class, dtype: int64
```

Evaluation (TP, FP, TN, FN)

```
In [14]: ▶ from sklearn.metrics import confusion_matrix
```

```
In [15]: ▶ confusion_matrix(y_test, predictions)
```

```
Out[15]: array([[2, 5],  
                [3, 8]], dtype=int64)
```

Confusion Matrix

		PREDICTIVE VALUES	
		POSITIVE (1)	NEGATIVE (0)
ACTUAL VALUES	POSITIVE (1)	TP	FN
	NEGATIVE (0)	FP	TN

```
array([[2, 5],  
       [3, 8]], dtype=int64)
```

Confusion Matrix Error I & Error II

$$\text{Accuracy} = (TP+TN)/(TP+FP+FN+TN) \quad \text{صحت}$$

$$\text{Precision} = TP/(TP+FP) \quad \text{دقت}$$

$$\text{Recall} = TP/(TP+FN) \quad \text{بازخوانی}$$

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

Accuracy Score

```
In [16]: ▶ from sklearn.metrics import accuracy_score
```

```
In [17]: ▶ accuracy_score(y_test, predictions, normalize=False)
```

```
Out[17]: 10
```

```
In [18]: ▶ accuracy_score(y_test, predictions, normalize=True)
```

```
Out[18]: 0.5555555555555556
```


Classification Report

```
In [19]: ▶ from sklearn.metrics import classification_report
```

```
In [20]: ▶ print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.40	0.29	0.33	7
1	0.62	0.73	0.67	11
accuracy			0.56	18
macro avg	0.51	0.51	0.50	18
weighted avg	0.53	0.56	0.54	18

Grid Search

```
▶ knn_1 = KNeighborsClassifier()
```

Init signature:

```
▶ KNeighborsClassifier(  
    n_neighbors=5,  
    *,  
    weights='uniform',  
    algorithm='auto',  
    leaf_size=30,  
    p=2,  
    metric='minkowski',  
    metric_params=None,
```

Grid Search

```
➤ knn_1 = KNeighborsClassifier()
```

```
➤ # Allows us to test parameters of classification algorithms and find the best one  
from sklearn.model_selection import GridSearchCV
```

```
➤ parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}
```


```
➤ knn_cv = GridSearchCV(knn_1, parameters)  
knn_cv.fit(X_train, y_train)
```

```
4]: GridSearchCV(estimator=KNeighborsClassifier(),  
                 param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]})
```

Tune Hpyerparameters

```
In [32]: ▶ print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
```

```
tuned hpyerparameters :(best parameters) {'n_neighbors': 9}
```



```
In [33]: ▶ knn_1 = KNeighborsClassifier(n_neighbors=9)
```

```
In [34]: ▶ knn_1.fit(X_train,y_train)
```

```
Out[34]: KNeighborsClassifier(n_neighbors=9)
```

Result

```
In [35]: ► predictions_1 = knn_1.predict(X_test)
```

```
In [36]: ► confusion_matrix(y_test, predictions_1)
```

```
Out[36]: array([[2, 5],  
               [2, 9]], dtype=int64)
```

```
In [37]: ► accuracy_score(y_test, predictions_1, normalize=False)
```

```
Out[37]: 11
```

```
In [38]: ► accuracy_score(y_test, predictions_1, normalize=True)
```

```
Out[38]: 0.6111111111111112
```

Assignment:

تمرین:

کدهای ارائه شده در درس را در نوتبوک جدیدی انجام داده و در صورت نیاز از نوتبوک هفته هفتم استفاده کنید.

برای تمرین بیشتر جلسه دهم دوره منتورینگ دیتاساینس را در کانال یوتیوب مشاهده کنید.

رزومه خود را آپدیت کرده و مهارتهایی که تا به امروز فراگرفته اید را اضافه کنید.